PRODUCT CODE: AC-9236F-MC
PRODUCT NAME: CZRKIFO RK11 UTILITY PACKAGE
DATE CREATED: MARCH, 1978
MAINTAINER: DIAGNOSTIC GROUP
AUTHOR: BOB COLLINS
REVISED BY: JIM KAPADIA
           TOM SANTER
           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 RKS5 CONTROL PANEL TEST
9.6 RKS5 CONTROL PANEL TEST # 2
9.7 HEAD ALIGNMENT ROUTINE
9.8 (DISK) POWER FAILURE TEST
9.9 SECTION SPECIAL
9.10 COMPATIBILITY ERROR RECOVERY
THIS PACKAGE CONTAINS 4 INDIVIDUAL UTILITY PROGRAMS FOR THE RKQX 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.

REQUIREMENTS

2 1 EQUIPMENT
PDP-11 PROCESSOR
8K MEMORY
RK11 OR RKV11 CONTROLLER
1-8 RKOS OR RKO$F 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

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.

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 17% (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 ' software 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:
TIME SWITCH SETTING ARE ENTERED, THE ENTIRE SWITCH REGISTER
IMAGE MUST BE ENTERED. LEADING ZEROS ARE NOT REQUIRED. 'RUBOUT' AND
'CTRL-AX'-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

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)

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)

RESTRICTIONS

7.1 STARTING RESTRICTIONS
IT IS NOT RECOMMENDED THAT YOU START AT AN
ADDRESS OTHER THAN ZEO, (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)

EXECUTION TIME
VARIATES WITH SELECTED ROUTINE, NUMBER OF DRIVES, ETC.

PROGRAM DESCRIPTION
THE RKII UTILITY PACKAGE IS DIVIDED INTO EIGHT SECTIONS WHICH ALLOW COMPATIBILITY TESTING, OSCILLATING SEEKER 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:

<table>
<thead>
<tr>
<th>SECTION</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>INDEX</td>
</tr>
<tr>
<td>1</td>
<td>COMPATIBILITY TEST</td>
</tr>
<tr>
<td>2</td>
<td>OSCILLATING SEEK PACKAGE</td>
</tr>
<tr>
<td>3</td>
<td>FORMATTER SURFACE VERIFIER</td>
</tr>
<tr>
<td>4</td>
<td>FRONT PANEL TEST</td>
</tr>
<tr>
<td>5</td>
<td>RKDS CONTROL PANEL TEST #2</td>
</tr>
<tr>
<td>6</td>
<td>HEAD ALIGNMENT ROUTINE</td>
</tr>
<tr>
<td>7</td>
<td>POWER FAILURE (DURING WRITE) TEST</td>
</tr>
</tbody>
</table>

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 REPEATING VARIOUS SWITCH SETTINGS.


USE: THIS IS EXAMPLE OF THE ACTUAL OUTPUT:

RKII UTILITY PACKAGE

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>INDEX</td>
<td>0</td>
</tr>
<tr>
<td>COMPATIBILITY PACKAGE</td>
<td>1</td>
</tr>
<tr>
<td>OSCILLATING SEEK PACKAGE</td>
<td>2</td>
</tr>
<tr>
<td>FORMATTER-SURFACE VERIFIER</td>
<td>3</td>
</tr>
<tr>
<td>RKDS CONTROL PANEL TEST #2</td>
<td>4</td>
</tr>
<tr>
<td>RKDS CONTROL PANEL TEST #2</td>
<td>5</td>
</tr>
<tr>
<td>HEAD ALIGNMENT ROUTINE</td>
<td>6</td>
</tr>
<tr>
<td>POWER FAILURE (WRITE) .EST</td>
<td>7</td>
</tr>
</tbody>
</table>

TYPE=x WHERE "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, THIS PACKAGE DOES NOT APPLY TO RK-DSF DRIVES.

DESCRIPTION: THIS PACKAGE ALLOWS A USER TO AUTOMATICALLY TEST
COMPATIBILITY OF UP TO EIGHT (8) DRIVES SIMPLY BY STATING THE DRIVE NUMBERS TO BE TESTED. THE TEST DOES NOT REQUIRE THE USER TO PLACE THE PACK. THE LIMITATIONS OF TESTING ARE IF THERE ARE (2) TWO PROCESSORS FROM ONE (1) TO SEVEN (7) DRIVE 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 TESTS COMPATIBILITY ON THREE (3) DRIVES WHOSE UNIT NUMBERS ARE 0, 1, 3....

**EXAMPLE 1**

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMPATIBILITY PACKAGE</td>
<td>0</td>
</tr>
<tr>
<td>OSCILLATING SEEK PACKAGE</td>
<td>1</td>
</tr>
<tr>
<td>FORMATTER-SURFACE VERIFIER</td>
<td>1</td>
</tr>
<tr>
<td>RKDS CONTROL PANEL TEST</td>
<td>1</td>
</tr>
<tr>
<td>RKDS CONTROL PANEL TEST #2</td>
<td>1</td>
</tr>
<tr>
<td>HEAD ALIGNMENT ROUTINE</td>
<td>1</td>
</tr>
<tr>
<td>POWER FAILURE (WRITE) TEST</td>
<td>1</td>
</tr>
</tbody>
</table>

TYPE=1

DRIVE NUMBERS ON SYSTEM 1=0, 1, 3.

IS THERE A SECOND SYSTEM?

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!
The user selected type one (1) and received the message RKX0 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....

RKII UTIL (PG) PACKAGE

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMPATIBILITY PACKAGE</td>
<td>0</td>
</tr>
<tr>
<td>OSCILLATING SEEK PACKAGE</td>
<td>1</td>
</tr>
<tr>
<td>FORMATER-SURFACE VERIFIER</td>
<td>2</td>
</tr>
<tr>
<td>RK05 CONTROL PANEL TEST</td>
<td>3</td>
</tr>
<tr>
<td>RK05 CONTROL PANEL TEST #2</td>
<td>4</td>
</tr>
<tr>
<td>HEAD ALIGNMENT ROUTINE</td>
<td>5</td>
</tr>
<tr>
<td>POWER FAILURE (WRITE) TEST</td>
<td>6</td>
</tr>
<tr>
<td>TYPE=1</td>
<td></td>
</tr>
</tbody>
</table>

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 ready mount pack on drive #0 make pack write enable press continue when drive ready 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 #0
DONE SYSTEM 2 RESTART SYSTEM 1, TYPE WORD 000077

**********
... THE USER RESPONSE TO THE QUESTION WORD 1 =

BY TYPING WORD 1 FROM PROCESOR 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 #0
DONE!

RK11 UTILITY PACKAGE

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

TYPE=

**********
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 SMD to SHP of the switch register. A switch is set for each drive (e.g. SMD to TEST DRIVES).

The user then inserts the address to seek in the SHP. If both addresses are legal 50 cycles (100 seeks) will be made between the specified address then the program will look at the SHP for possible change. This should allow for good stable tracks on an oscilloscope.

It should be noted that the oscillating seeks between the specified cylinders are done on all available drives. The only way to exit is hail!, load address 200, hit start.

USE: Select type 2, respond to question with unit number...

OSCILLATING SEEK PACKAGE

SET SMD TO SHP 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 (bit 0-7) of the switch register and the "Last Cylinder Address" (Inner Limit) into the high byte (bit 8-15). Then press continue.

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 R&F-5F should not be selected for testing at the same time.

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.
MESSAGE IS TYPED AND PACK IS FORMATTED.
SELECT TYPE 3. RESPOND TO QUESTION WITH
SETTING OF SWITCH REGISTER.

RK11 UTILITY PACKAGE

INDEX NAME TYPE
0 COMPATIBILITY PACKAGE
1

Oscillating Seek Package
Formatter-Surface Verifier
RKOS Control Panel Test
RKOS Control Panel Test #2
Head Alignment Routine
Power Failure (Write) Test

TYPE=3
Formatter-Surface Verifier, Set SW REG WITH DRIVE # 1's
PACK GOOD.

RK11 UTILITY PACKAGE

INDEX NAME TYPE
0 COMPATIBILITY PACKAGE
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

SECTION 4 RKOS Control Panel Test

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

INDEX NAME TYPE
0 COMPATIBILITY PACKAGE
1

Oscillating Seek Package
Formatter-Surface Verifier
RKOS Control Panel Test
RKOS Control Panel Test #2
Head Alignment Routine
Power Failure (Write) Test

TYPE=4
RXOS CONTROL PANEL TEST WHICH DRIVES MOUNT PAK ON DRIVE A
PLACE DRIVE IN RUN SHOULD SEE THE RUN, POWER AND CYLINDER LAMPS LIGHT.
WRITE PROTECT 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 DUNE!

RX11 UTILITY PACKAGE

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMPATIBILITY PACKAGE</td>
<td>0</td>
</tr>
<tr>
<td>OSCILLATING SEEK PACKAGE</td>
<td>3</td>
</tr>
<tr>
<td>FORMATTER-SPURFACE VERIFIER</td>
<td>4</td>
</tr>
<tr>
<td>RXOS CONTROL PANEL TEST</td>
<td>5</td>
</tr>
<tr>
<td>RXOS CONTROL PANEL TEST #2</td>
<td>6</td>
</tr>
<tr>
<td>HEAD ALIGNMENT ROUTINE</td>
<td></td>
</tr>
<tr>
<td>POWER FAILURE (WRITE) TEST</td>
<td></td>
</tr>
</tbody>
</table>

************

9.6 SECTION 5 RXOS 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

A software program then starts scanning all the 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 data is 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. If seek incomplete occurs when any change in the status is found, it is reported. If the drive is 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 LD
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 SHd=0 FOR SURFACE 0, SHd=1 FOR SURFACE 1.
SET SM=1 TO TEST CYL 64, SET SM=0 TO TEST CYLINDER IDS.
SM=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 H/TO TYPE drives.
THE UPPER OR THE LOWER HEAD CAN BE
SELECTED BY SWITCH O. IF SURFACE O IS
TO BE SELECTED, PUT SW O TO O. IF
SURFACE 1 IS TO BE SELECTED, PUT SW 1 ON 1.
THE HEADS MAY BE POSITIONED AT CYLINDER 64
OR CYLINDER 105. SET SW 0 FOR CYLINDER 105.
SW 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 1 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-DSJ CARTRIDGE
SO IF AN F TYPE DRIVE IS SELECTED, CYLINDER
64 OF THE RK-DSJ IS CYLINDER 130 OF THE F DRIVE
(EVEN DRIVE), CYLINDER 105 BECOMES CYLINDER 5
OF THE ODD DRIVE ON THE RK-DSJ.

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 NUMBER 0, 1, 2, 3, 4, 5, 6, 7"
THEN IT PROCEEDS TO WRITE UNIQUE PATTERNS
ON CYLINDERS 0 TO 15 (DECIMAL) OF THAT DRIVE.
THE HEADS ARE THEN POSITIONED ON CYLINDER 10
AND THE USER IS ASKED TO DROP POWER ON THAT
DRIVE:
"DROP POWER"
MEANWHILE WRITE IS BEING DONE ON CYLINDER 10.
ON RECEIVING 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 RKDR=XXXX"
XXXX IS THE CONTENTS OF RKDR 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 MALT.

### SECTION SPECIAL

FOR THE BELOW EXAMPLES THE FOLLOWING FORMAT WILL BE USED:

- **TYPE**
- **INDEX**
- **NAME**
- **COMMANDS**
- **ERROR EXAMPLE 1**

<table>
<thead>
<tr>
<th>INDEX</th>
<th>NAME</th>
<th>TYPE</th>
<th>COMMANDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>RIKO CONTROL PANEL TEST</td>
<td>4</td>
<td>TYPE 1 SELECTION</td>
</tr>
<tr>
<td>0</td>
<td>RIKO CONTROL PANEL TEST</td>
<td>1</td>
<td>DRIVE #0 SELECTED</td>
</tr>
<tr>
<td>1</td>
<td>RIKO CONTROL PANEL TEST</td>
<td>2</td>
<td>IS THERE A SECOND SYSTEM?</td>
</tr>
<tr>
<td>2</td>
<td>RIKO CONTROL PANEL TEST</td>
<td>3</td>
<td>MAKE PACK WRITE ENABLE</td>
</tr>
<tr>
<td>4</td>
<td>RIKO CONTROL PANEL TEST</td>
<td>4</td>
<td>PRESS CONTINUE WHEN DRIVE NOT READY</td>
</tr>
<tr>
<td>5</td>
<td>RIKO CONTROL PANEL TEST</td>
<td>5</td>
<td>DRIVE WRITE PROTECTED</td>
</tr>
<tr>
<td>6</td>
<td>RIKO CONTROL PANEL TEST</td>
<td>6</td>
<td>DONE!</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>INDEX</th>
<th>NAME</th>
<th>TYPE</th>
<th>COMMANDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>COMPATIBILITY PACKAGE</td>
<td>0</td>
<td>NAME</td>
</tr>
<tr>
<td>1</td>
<td>OSCILLATING SEEK PACKAGE</td>
<td>1</td>
<td>TYPE 1 SELECTION</td>
</tr>
</tbody>
</table>

ERROR EXAMPLE 2

<table>
<thead>
<tr>
<th>INDEX</th>
<th>NAME</th>
<th>TYPE</th>
<th>COMMANDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>COMPATIBILITY PACKAGE</td>
<td>0</td>
<td>NAME</td>
</tr>
<tr>
<td>1</td>
<td>OSCILLATING SEEK PACKAGE</td>
<td>1</td>
<td>TYPE 1 SELECTION</td>
</tr>
<tr>
<td>2</td>
<td>RIKO CONTROL PANEL TEST</td>
<td>2</td>
<td>DRIVE #0 SELECTED</td>
</tr>
<tr>
<td>3</td>
<td>RIKO CONTROL PANEL TEST</td>
<td>3</td>
<td>IS THERE A SECOND SYSTEM?</td>
</tr>
<tr>
<td>4</td>
<td>RIKO CONTROL PANEL TEST</td>
<td>4</td>
<td>MAKE PACK WRITE ENABLE</td>
</tr>
<tr>
<td>5</td>
<td>RIKO CONTROL PANEL TEST</td>
<td>5</td>
<td>PRESS CONTINUE WHEN DRIVE NOT READY</td>
</tr>
<tr>
<td>6</td>
<td>RIKO CONTROL PANEL TEST</td>
<td>6</td>
<td>DRIVE WRITE PROTECTED</td>
</tr>
<tr>
<td>7</td>
<td>RIKO CONTROL PANEL TEST</td>
<td>7</td>
<td>DONE!</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>INDEX</th>
<th>NAME</th>
<th>TYPE</th>
<th>COMMANDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>COMPATIBILITY PACKAGE</td>
<td>0</td>
<td>NAME</td>
</tr>
<tr>
<td>1</td>
<td>OSCILLATING SEEK PACKAGE</td>
<td>1</td>
<td>TYPE 1 SELECTION</td>
</tr>
</tbody>
</table>

ERROR EXAMPLE 3

<table>
<thead>
<tr>
<th>INDEX</th>
<th>NAME</th>
<th>TYPE</th>
<th>COMMANDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>COMPATIBILITY PACKAGE</td>
<td>0</td>
<td>NAME</td>
</tr>
<tr>
<td>1</td>
<td>OSCILLATING SEEK PACKAGE</td>
<td>1</td>
<td>TYPE 1 SELECTION</td>
</tr>
</tbody>
</table>

ERROR EXAMPLE 4

<table>
<thead>
<tr>
<th>INDEX</th>
<th>NAME</th>
<th>TYPE</th>
<th>COMMANDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>COMPATIBILITY PACKAGE</td>
<td>0</td>
<td>NAME</td>
</tr>
<tr>
<td>1</td>
<td>OSCILLATING SEEK PACKAGE</td>
<td>1</td>
<td>TYPE 1 SELECTION</td>
</tr>
</tbody>
</table>

ERROR EXAMPLE 5

<table>
<thead>
<tr>
<th>INDEX</th>
<th>NAME</th>
<th>TYPE</th>
<th>COMMANDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>COMPATIBILITY PACKAGE</td>
<td>0</td>
<td>NAME</td>
</tr>
<tr>
<td>1</td>
<td>OSCILLATING SEEK PACKAGE</td>
<td>1</td>
<td>TYPE 1 SELECTION</td>
</tr>
</tbody>
</table>

ERROR EXAMPLE 6

<table>
<thead>
<tr>
<th>INDEX</th>
<th>NAME</th>
<th>TYPE</th>
<th>COMMANDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>COMPATIBILITY PACKAGE</td>
<td>0</td>
<td>NAME</td>
</tr>
<tr>
<td>1</td>
<td>OSCILLATING SEEK PACKAGE</td>
<td>1</td>
<td>TYPE 1 SELECTION</td>
</tr>
</tbody>
</table>

ERROR EXAMPLE 7

<table>
<thead>
<tr>
<th>INDEX</th>
<th>NAME</th>
<th>TYPE</th>
<th>COMMANDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>COMPATIBILITY PACKAGE</td>
<td>0</td>
<td>NAME</td>
</tr>
<tr>
<td>1</td>
<td>OSCILLATING SEEK PACKAGE</td>
<td>1</td>
<td>TYPE 1 SELECTION</td>
</tr>
</tbody>
</table>

ERROR EXAMPLE 8

<table>
<thead>
<tr>
<th>INDEX</th>
<th>NAME</th>
<th>TYPE</th>
<th>COMMANDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>COMPATIBILITY PACKAGE</td>
<td>0</td>
<td>NAME</td>
</tr>
<tr>
<td>1</td>
<td>OSCILLATING SEEK PACKAGE</td>
<td>1</td>
<td>TYPE 1 SELECTION</td>
</tr>
</tbody>
</table>

ERROR EXAMPLE 9

<table>
<thead>
<tr>
<th>INDEX</th>
<th>NAME</th>
<th>TYPE</th>
<th>COMMANDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>COMPATIBILITY PACKAGE</td>
<td>0</td>
<td>NAME</td>
</tr>
<tr>
<td>1</td>
<td>OSCILLATING SEEK PACKAGE</td>
<td>1</td>
<td>TYPE 1 SELECTION</td>
</tr>
</tbody>
</table>
CO2

DRIVE NOT READY : RUN IF NOT LOADED OR NOT READY MAKING DRIVE READY WILL STOP THE MESSAGE
DRIVE NOT READY : IT DOES NOT EFFECT THE OUTCOME OF COMPATIBILITY

RKII UTILITY PACKAGE

NAME TYPE
INDEX 0
COMPATIBILITY PACKAGE 1
OSCILLATING SEEK PACKAGE 2

ERROR EXAMPLE 3

RKII UTILITY PACKAGE

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

TYPE=1
DRIVE NUMBERS ON SYSTEM 1=0,4,7

IS THERE A SECOND SYSTEM?
DRIVE #2
MOUNT PACK ON DRIVE #2
MAKE PACK WRITE ENABLE
PRESS CONTINUE WHEN DRIVE RDY
MOUNT PACK ON DRIVE #1
MAKE PACK WRITE ENABLE
PRESS CONTINUE WHEN DRIVE RDY
DRIVE RESET TIMED OUT
DRIVE RESET TIMED OUT
DRIVE RESET TIMED OUT
DRIVE RESET TIMED OUT

#NOTE A SLOW DRIVE OR FAST PROCESSOR AND MEMORY MAY CAUSE THE MESSAGE TO APPEAR A FEW TIMES AND THEN CONTINUE THIS IS OK AND WILL NOT EFFECT THE OUTCOME OF THE TEST.

ERROR EXAMPLE 4

OSCILLATING SEEK PACKAGE 0
FORMATTED-SURFACE VERIFIER 1
RKOS CONTROL PANEL TEST 2

TYPE=1
ERROR EXAMPLE 5

THE BELOW ERRORS DO NOT ALWAYS EFFECT COMPATIBILITY.
IN THE FIRST TYPE THE DRIVE IS DOWN INDICATING THAT (S) 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 COMPATIBILITY GOES. THAT IS TO SAY IT IS NOT TESTED, THEREFORE NOT NECESSARILY COMPATIBLE OR INCOMPATIBLE.

INDEX
NAME
TYPE
COMPARABILITY PACKAGE
0
OCCILLATING SEEK PACKAGE
1
FORMATTING SUB PACKAGE
2
AKOS CONTROL PANEL TEST
3
AKS CONTROL PANEL TEST #2
4
HEAD ALIGNMENT ROUTINE
5
POWER FAILURE (WRITE) TEST
6

*TYPE 1

DRIVE NUMBERS ON SYTEM 1=0.

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

NAME
TYPE
INDEX

ERROR EXAMPLE 6

*IN THE ABOVE CASE THE MESSAGE "J 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 J SEEK ERRORS OCCURRING
ON ONE DRIVE.
RKII UTILITY PACKAGE

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

TYPE = 1
DRIVE NUMBERS ON SYSTEM I = 0.
IS THERE A SECOND SYSTEM?
DRIVE I = 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 I CANNOT BE READ.
ADRR = 007641  EXPECT = 077400  RECV = 177000
ADRR = 007644  EXPECT = 077400  RECV = 177000
ADRR = 007647  EXPECT = 077400  RECV = 177000
ADRR = 007650  EXPECT = 077400  RECV = 177000
ERROR! DATA WRITTEN BY DRIVE I CANNOT BE READ.
ADRR = 007653  EXPECT = 077400  RECV = 177000
ADRR = 007656  EXPECT = 077400  RECV = 177000
DONE:
RKII UTILITY PACKAGE

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

TYPE = 1
THE ABOVE ERROR MESSAGE SHOWS A COMPATIBILITY
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.
PRESS CONTINUE WHEN DRIVE READY.
ERROR! Data written by drive 1 cannot be read.
ADDRESS: 00077600 00077600 00077600 00077600
ADDRESS: 00077700 00077700 00077700 00077700
ADDRESS: 00077800 00077800 00077800 00077800
ADDRESS: 00077900 00077900 00077900 00077900
ADDRESS: 00077A00 00077A00 00077A00 00077A00
ADDRESS: 00077B00 00077B00 00077B00 00077B00
ADDRESS: 00077C00 00077C00 00077C00 00077C00
ADDRESS: 00077D00 00077D00 00077D00 00077D00
ADDRESS: 00077E00 00077E00 00077E00 00077E00
ADDRESS: 00077F00 00077F00 00077F00 00077F00
ADDRESS: 00078000 00078000 00078000 00078000
ADDRESS: 00078100 00078100 00078100 00078100
ADDRESS: 00078200 00078200 00078200 00078200
ADDRESS: 00078300 00078300 00078300 00078300
ADDRESS: 00078400 00078400 00078400 00078400
ADDRESS: 00078500 00078500 00078500 00078500
ADDRESS: 00078600 00078600 00078600 00078600
ADDRESS: 00078700 00078700 00078700 00078700
ERROR! Data written by drive 1 cannot be read.
ADDRESS: 00027600 00027600 00027600 00027600
ERRORS OCCURRED AT 10:25 0000 REC'D 1770000.
DUE:

In the above example, the problem is extreme.
The drive was declared down due to checksum errors. (To see how this was determined see Paragraph 9.7.) Notice also the problem did not appear until cylinder 57, and was not fatal until cylinder 57, an odd head #1 was a common factor.

******

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 receive more information or control parameters.

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 executing. The second is in the 'error' routine which allows the user to examine the error register before the program corrects any errors which may have occurred.

If placed by checksum errors and the user wishes more error masking, then he may modify the mask word at location 'errchk=2' to only recognize hard errors.

To increase or decrease the number of retries allowed
BEFORE A DRIVE IS DECLARED DOWN, GO TO THE
'MOUNT' ROUTINE, MODIFY THE SETUP OF LOCATIONS
'ECON' AND 'CMTSIN' AND YOU HAVE IT!

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 'CHCEN' IN THE 'RDCHK' ROUTINE.

A FINAL LOOK: THE FOLLOWING SECTION SHOWS ALL PACKAGES
CALLED IN SEQUENCE, NONE WITH ERRORS.

AKII UTILITY PACKAGE

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMPATIBILITY PACKAGE</td>
<td>0</td>
</tr>
<tr>
<td>OSCILLATING SEEK PACKAGE</td>
<td>1</td>
</tr>
<tr>
<td>FORMATTER-SURFACE VERIFIER</td>
<td>2</td>
</tr>
<tr>
<td>RKOS CONTROL PANEL TEST</td>
<td>3</td>
</tr>
<tr>
<td>RKOS CONTROL PANEL TEST #2</td>
<td>4</td>
</tr>
<tr>
<td>HEAD ALIGNMENT ROUTINE</td>
<td>5</td>
</tr>
<tr>
<td>POWER FAILURE (WRITE) TEST</td>
<td>6</td>
</tr>
</tbody>
</table>

AKII UTILITSY PACKAGE

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMPATIBILITY PACKAGE</td>
<td>0</td>
</tr>
<tr>
<td>OSCILLATING SEEK PACKAGE</td>
<td>1</td>
</tr>
<tr>
<td>FORMATTER-SURFACE VERIFIER</td>
<td>2</td>
</tr>
<tr>
<td>RKOS CONTROL PANEL TEST</td>
<td>3</td>
</tr>
<tr>
<td>RKOS CONTROL PANEL TEST #2</td>
<td>4</td>
</tr>
<tr>
<td>HEAD ALIGNMENT ROUTINE</td>
<td>5</td>
</tr>
<tr>
<td>POWER FAILURE (WRITE) TEST</td>
<td>6</td>
</tr>
</tbody>
</table>

TYPE=0

DRIVE NUMBERS ON SYSTEM 1=0,1,3.

IS THERE A SECOND SYSTEM?
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 #2
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 #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**

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMPATIBILITY PACKAGE</td>
<td>0</td>
</tr>
<tr>
<td>OSCILLATING SEEK PACKAGE</td>
<td>1</td>
</tr>
<tr>
<td>FORMATTER-SURFACE VERIFIER</td>
<td>2</td>
</tr>
<tr>
<td>RKOS CONTROL PANEL TEST</td>
<td>3</td>
</tr>
<tr>
<td>RKOS CONTROL PANEL TEST #2</td>
<td>4</td>
</tr>
<tr>
<td>HEAD ALIGNMENT ROUTINE</td>
<td>5</td>
</tr>
<tr>
<td>POWER FAILURE (WRITE) TEST</td>
<td>6</td>
</tr>
</tbody>
</table>

**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**

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMPATIBILITY PACKAGE</td>
<td>0</td>
</tr>
<tr>
<td>OSCILLATING SEEK PACKAGE</td>
<td>1</td>
</tr>
<tr>
<td>FORMATTER-SURFACE VERIFIER</td>
<td>2</td>
</tr>
<tr>
<td>RKOS CONTROL PANEL TEST</td>
<td>3</td>
</tr>
<tr>
<td>RKOS CONTROL PANEL TEST #2</td>
<td>4</td>
</tr>
<tr>
<td>HEAD ALIGNMENT ROUTINE</td>
<td>5</td>
</tr>
<tr>
<td>POWER FAILURE (WRITE) TEST</td>
<td>6</td>
</tr>
</tbody>
</table>

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

**RK11 UTILITY PACKAGE**

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMPATIBILITY PACKAGE</td>
<td>0</td>
</tr>
<tr>
<td>OSCILLATING SEEK PACKAGE</td>
<td>1</td>
</tr>
<tr>
<td>FORMATTER-SURFACE VERIFIER</td>
<td>2</td>
</tr>
<tr>
<td>RKOS CONTROL PANEL TEST</td>
<td>3</td>
</tr>
<tr>
<td>RKOS CONTROL PANEL TEST #2</td>
<td>4</td>
</tr>
<tr>
<td>HEAD ALIGNMENT ROUTINE</td>
<td>5</td>
</tr>
<tr>
<td>POWER FAILURE (WRITE) TEST</td>
<td>6</td>
</tr>
</tbody>
</table>

**TYPE=4**
RKOS CONTROL PANEL TEST, WHICH DRIVE?0
MOUNT PACK ON DRIVE?0
PLACE DRIVE IN RUN, SHOULD SEE THE RUN,
POWER, AND 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

P - 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
MAINIC-11-0Z81I-E  MACY11 30A(1052)  24-MAR-78  09:23
0Z81I P11  24-MAR-78  09:20  TABLE OF CONTENTS

22  BASIC DEFINITIONS
132  TRAP CATCHER
141  STARTING ADDRESS(ES)
146  ACTII HOOKS
156  COMMON TAGS
296  ERROR POINTER TABLE
324  INITIALIZE THE COMMON TAGS
363  TYPE PROGRAM NAME
368  GET VALUE FOR SOFTWARE SWITCH REGISTER
440  COMPATIBILITY TEST
1153  OSCILLATING SEEK ROUTINE
1397  FORMATTER-SURFACE VERIFIER
1571  AGOS CONTROL PANEL TEST
1841  CONTROL PANEL TEST # 2
2185  HEAD ALIGNMENT ROUTING
2288  DISK POWER FAILURE TEST
2412  TYPE ROUTINE
2486  BINARY TO OCTAL (ASCII) AND TYPE
2573  TTY INPUT ROUTINE
2799  READ AN OCTAL NUMBER FROM THE TTY
2837  TRAP DECODER
2860  TRAP TABLE
.TITLE MAINDEC-11-DZRK1-E
"COPYRIGHT (C) 1974, 1977
"DIGITAL EQUIPMENT CORP.
"RAYNARD, MASS. 01754
"PROGRAM BY BOB COLLINS
"THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYMMAC
"REVISED BY JIM KAPADIA
"REVISED BY TOM SAWYER FEB 27, 1976
"REVISED BY CHUCK HESS AUGUST, 1976
"INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
"STACK= 1100
"EQUIV EXT.ERR = BASIC DEFINITION OF ERROR CALL
"EQUIV IOT,SCOPE = BASIC DEFINITION OF SCOPE CALL
"MISCELLANEOUS DEFINITIONS
HT= 11 CODE FOR HORIZONTAL TAB
LF= 15 CODE FOR LINE FEED
CR= 20 CODE FOR CARRIAGE RETURN
CRLF= 200 CODE FOR CARRIAGE RETURN-LINE FEED
PS= 177776 PROCESSOR STATUS WORD
P_EQUIV PS,PSW
STKLM= 177774 STACK LIMIT REGISTER
PIR= 177774 PROGRAM INTERRUPT REQUEST REGISTER
DSW= 177570 HARDWARE SWITCH REGISTER
DDISP= 177570 HARDWARE DISPLAY REGISTER
"GENERAL PURPOSE REGISTER DEFINITIONS
AD= &0 GENERAL REGISTER
AI= &1 GENERAL REGISTER
AR= &2 GENERAL REGISTER
AR= &3 GENERAL REGISTER
RH= &4 GENERAL REGISTER
RG= &5 GENERAL REGISTER
RB= &6 GENERAL REGISTER
RT= &7 GENERAL REGISTER
SP= &8 STACK POINTER
PC= &9 PROGRAM COUNTER
"PRIORITY LEVEL DEFINITIONS
PRD= &0 PRIORITY LEVEL 0
PRI= &40 PRIORITY LEVEL 1
PRL= &100 PRIORITY LEVEL 2
PR2= &140 PRIORITY LEVEL 3
PR4= &200 PRIORITY LEVEL 4
::"SWITCH REGISTER" SWITCH DEFINITIONS

SWI5 = 10000
SWI4 = 4000
SWI3 = 2000
SWI2 = 1000
SWI1 = 400
SWI0 = 100
SWO5 = 200
SWO4 = 50
SWO3 = 10
SWO2 = 5
SWO1 = 1
SWO0 = 1

EQUIV SWI5 = 00001
EQUIV SWI4 = 000001
EQUIV SWI3 = 0000001
EQUIV SWI2 = 00000001
EQUIV SWI1 = 000000001
EQUIV SWI0 = 0000000001
EQUIV SWO5 = 00000000001
EQUIV SWO4 = 000000000001
EQUIV SWO3 = 0000000000001
EQUIV SWO2 = 00000000000001
EQUIV SWO1 = 000000000000001
EQUIV SWO0 = 0000000000000001

::DATA BIT DEFINITIONS (BIT00 TO BIT15)

BIT15 = 10000
BIT14 = 4000
BIT13 = 2000
BIT12 = 1000
BIT11 = 400
BIT10 = 200
BIT09 = 100
BIT08 = 40
BIT07 = 20
BIT06 = 10
BIT05 = 4
BIT04 = 2
BIT03 = 1
BIT02 = 0
BIT01 = 0
BIT00 = 0

EQUIV BIT09.BIT0
EQUIV BIT08.BIT07
EQUIV BIT07.BIT06
EQUIV BIT06.BIT05
EQUIV BIT05.BIT04
EQUIV BIT04.BIT03
EQUIV BIT03.BIT02
EQUIV BIT02.BIT01
EQUIV BIT01.BIT00
EQUIV BIT0,BIT1
EQUIV BIT0,BIT0

**BASIC "CPU" TRAP VECTOR ADDRESSES**

ERRVEC = 4
RESVEC = 10
TBivec = 14
TRTvec = 14
BPTvec = 14
IOTvec = 20
PURVEC = 54
ENTVEC = 30
TRAPVEC = 34
TKVEC = 60
TPVEC = 64
PIRVEC = 240
SBTTL TRAP CATCHER

0

**ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ",+2,HALT"**

**SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS**

**LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS**

DISPREG: WORD 0
SREG: WORD 0
SBTTL STARTING ADDRESS(ES)
JMP #STARTR

**SOFTWARE DISPLAY REGISTER**

**SOFTWARE SWITCH REGISTER**

MOV #377,bmode
JMP #START

**SBTTL ACT11 HOOKS**

**HOOKS REQUIRED BY ACT11**

$VPC=

.SAVE PC

$ENDAD

;1) SET LOC.46 TO ADDRESS OF $ENDAD IN $EOP

.WORD 0

;2) SET LOC.52 TO ZERO
**COMMON TAGS**

---

**NOTE:**
- This table contains various common storage locations used in the program.

<table>
<thead>
<tr>
<th>TAG</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCONTAG</td>
<td>$1100: Start of common tags</td>
</tr>
<tr>
<td>SPASS</td>
<td>Contains pass count</td>
</tr>
<tr>
<td>SSTEPN</td>
<td>Contains step number</td>
</tr>
<tr>
<td>SHLET</td>
<td>Contains loop exit</td>
</tr>
<tr>
<td>HLET</td>
<td>Contains loop error</td>
</tr>
<tr>
<td>HPROC</td>
<td>Contains Proc loop</td>
</tr>
<tr>
<td>ERSET</td>
<td>Contains error set flag</td>
</tr>
<tr>
<td>ITM</td>
<td>Contains item control byte</td>
</tr>
<tr>
<td>IDEMA</td>
<td>Contains error message</td>
</tr>
<tr>
<td>SDEC</td>
<td>Contains index</td>
</tr>
<tr>
<td>SDAT</td>
<td>Contains data</td>
</tr>
<tr>
<td>BOUTB</td>
<td>Contains automatic mode</td>
</tr>
<tr>
<td>BINTAG</td>
<td>Contains interrupt mode</td>
</tr>
<tr>
<td>SWR</td>
<td>Address of switch register</td>
</tr>
<tr>
<td>DISPM</td>
<td>Address of display register</td>
</tr>
<tr>
<td>IXK</td>
<td>TTY keyboard status reg.</td>
</tr>
<tr>
<td>IBK</td>
<td>TTY keyboard buffer</td>
</tr>
<tr>
<td>IPS</td>
<td>TTY printer status reg.</td>
</tr>
<tr>
<td>IPBS</td>
<td>TTY printer buffer reg.</td>
</tr>
<tr>
<td>SFILL</td>
<td>Contains fill character</td>
</tr>
<tr>
<td>SFLEN</td>
<td>Contains fill length</td>
</tr>
<tr>
<td>SQUES</td>
<td>Contains question mark</td>
</tr>
<tr>
<td>SFF1</td>
<td>Contains string</td>
</tr>
<tr>
<td>DLFC</td>
<td>Contains line feed</td>
</tr>
<tr>
<td>DAHRCT</td>
<td>Contains active drive word</td>
</tr>
<tr>
<td>LOGA</td>
<td>Table of active drive words</td>
</tr>
<tr>
<td>DRV0</td>
<td>Table of pattern = to drive #'s</td>
</tr>
<tr>
<td>RDTBL</td>
<td>Table of read address</td>
</tr>
</tbody>
</table>
; TABLE OF CYLINDER BASE FOR AUTO MODE

BASE: .BYTE 0

; CYL 0 BASE CYLINDER ADDRESS
; CYL 10 BASE CYLINDER ADDRESS
; CYL 20 BASE CYLINDER ADDRESS
; CYL 30 BASE CYLINDER ADDRESS
; CYL 40 BASE CYLINDER ADDRESS
; CYL 50 BASE CYLINDER ADDRESS
; CYL 60 BASE CYLINDER ADDRESS
; CYL 70 BASE CYLINDER ADDRESS
; CYL 80 BASE CYLINDER ADDRESS
; CYL 90 BASE CYLINDER ADDRESS
; CYL 100 BASE CYLINDER ADDRESS

CYLCTL: .BLKB 11

; TABLE OF SELECTED BASES

SECTBL: .BYTE 0

; SECTOR 0
; SECTOR 1
; SECTOR 2
; SECTOR 3
; SECTOR 4
; SECTOR 5
; SECTOR 6
; SECTOR 7
; SECTOR 8
; SECTOR 9
; SECTOR 10
; SECTOR 11
; SECTOR 12

DRCNT1: .BYTE 0

; COUNT OF NUMBER OF DRIVES ON SYS. 1
; COUNT OF NUMBER OF DRIVES ON SYS. 2
; COUNT OF NUMBER OF IMMEDIATE DRIVES

PRONUM: .BYTE 0

; IF 0.1 PROCESSOR SELECTED
; IF 0.2 PROCESSOR SELECTED
; IF 0.3 PROCESSOR SELECTED

DRIVE: .BYTE 0

; DRIVE 0 UNDER TEST (MANUAL MODE)
; DRIVE 1 UNDER TEST (AUTO MODE)
; DRIVE 2 UNDER TEST (MANUAL MODE)
; DRIVE 3 UNDER TEST (AUTO MODE)

COMP: .BYTE 0

; BASE SELECTED (MANUAL MODE)
; BASE SELECTED (AUTO MODE)

WRTNBL: .BYTE 0

; DRIVE WHICH DID WRITE (READ OPERATION)
; DRIVE WHICH DID WRITE (WRITE OPERATION)
; DRIVE WHICH DID WRITE (READ/WRITE OPERATION)

ERROCCNT: .BYTE 0

; ERROR COUNTER
; ERROR COUNTER
; ERROR COUNTER

SCLTNCNT: .BYTE 0

; SEEK COUNT COUNTER
; SEEK COUNT COUNTER
; SEEK COUNT COUNTER

HIDAY: .BYTE 0

; SECOND PASS TIMER
; SECOND PASS TIMER
; SECOND PASS TIMER

STGFLG: .BYTE 0

; CURRENT INDEX #
; CURRENT INDEX #
; CURRENT INDEX #

KEYTEMP: .WORD 0

; TEMP. KEYBOARD BUFFER
; TEMP. KEYBOARD BUFFER
; TEMP. KEYBOARD BUFFER

CONTR: .WORD 0

; TEMP. CONTROL & STATUS WORD
; TEMP. CONTROL & STATUS WORD
; TEMP. CONTROL & STATUS WORD

GWR: .WORD 0

; TEMPERATURE & REGISTERS WORD
; TEMPERATURE & REGISTERS WORD
; TEMPERATURE & REGISTERS WORD

WBLCNT: .WORD 0

; TEMPERATURE & REGISTERS WORD
; TEMPERATURE & REGISTERS WORD
; TEMPERATURE & REGISTERS WORD

CMLCNT: .WORD -6000

; TEMPERATURE & REGISTERS WORD
; TEMPERATURE & REGISTERS WORD
; TEMPERATURE & REGISTERS WORD

SCTLCNT: .WORD -400

; TEMPERATURE & REGISTERS WORD
; TEMPERATURE & REGISTERS WORD
; TEMPERATURE & REGISTERS WORD

TYR: .WORD 0

; TEMPERATURE & REGISTERS WORD
; TEMPERATURE & REGISTERS WORD
; TEMPERATURE & REGISTERS WORD

SHRNCNT: .WORD 0

; TEMPERATURE & REGISTERS WORD
; TEMPERATURE & REGISTERS WORD
; TEMPERATURE & REGISTERS WORD

RESNCNT: .WORD 0

; TEMPERATURE & REGISTERS WORD
; TEMPERATURE & REGISTERS WORD
; TEMPERATURE & REGISTERS WORD

TAMCSS: .WORD 0

; TEMPERATURE & REGISTERS WORD
; TEMPERATURE & REGISTERS WORD
; TEMPERATURE & REGISTERS WORD

RECEC: .WORD 0

; TEMPERATURE & REGISTERS WORD
; TEMPERATURE & REGISTERS WORD
; TEMPERATURE & REGISTERS WORD

CNRNCNT: .WORD 0

; TEMPERATURE & REGISTERS WORD
; TEMPERATURE & REGISTERS WORD
; TEMPERATURE & REGISTERS WORD

TIM: .WORD 0

; TEMPERATURE & REGISTERS WORD
; TEMPERATURE & REGISTERS WORD
; TEMPERATURE & REGISTERS WORD

SAP: .WORD 0

; TEMPERATURE & REGISTERS WORD
; TEMPERATURE & REGISTERS WORD
; TEMPERATURE & REGISTERS WORD

WRITCS: .WORD 0

; TEMPERATURE & REGISTERS WORD
; TEMPERATURE & REGISTERS WORD
; TEMPERATURE & REGISTERS WORD

VID: .WORD 0

; TEMPERATURE & REGISTERS WORD
; TEMPERATURE & REGISTERS WORD
; TEMPERATURE & REGISTERS WORD

WRITD: .WORD 0

; TEMPERATURE & REGISTERS WORD
; TEMPERATURE & REGISTERS WORD
; TEMPERATURE & REGISTERS WORD

READS: .WORD 0

; TEMPERATURE & REGISTERS WORD
; TEMPERATURE & REGISTERS WORD
; TEMPERATURE & REGISTERS WORD

CONTENT: .WORD 0

; TEMPERATURE & REGISTERS WORD
; TEMPERATURE & REGISTERS WORD
; TEMPERATURE & REGISTERS WORD

PATT: .WORD 0

; TEMPERATURE & REGISTERS WORD
; TEMPERATURE & REGISTERS WORD
; TEMPERATURE & REGISTERS WORD

; DATA PATTERN
:BIT DEFINITIONS

DPL=BIT12
RMS=BIT8
WPS=BIT5
```
.SBTLT ERROR POINTER TABLE

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

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

$ERRTB:

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

;THE ERROR TABLE IS UNUSED IN THIS PROGRAM

;*****************************************************************
```
START: CLR  D8
:MODE  CLEAR THE BUS

START: MOV  $STACK_SP
:STACK SP
MOV  @D0,-(SP)  SET UP STACK FOR PSW=0
MOV  $25,-(SP)  RETURN FOR RTI

25: NOP

:INITIALIZE THE COMMON TAGS
::CLEAR COMMON TAGS ($COMMON) AREA
MOV  $COMMON,AB  FIRST LOCATION TO BE CLEARED
CLR  (AB)  CLEAR MEMORY LOCATION
CMP  @D0,AB  DONE?
BNE  -6  LOOP BACK IF NO

::INITIALIZE A FEW VECTORS
MOV  @D0,$TRAPVEC  TRAP VECTOR FOR TRAP CALLS
MOV  $END, @D0,$TRAPVEC  LEVEL 7
MOV  $END, @D0,$HARDWAREVECTOR  POWER FAILURE VECTOR
MOV  $END, @D0,$HARDWAREVECTOR  = LEVEL 7

::SIZE FOR A HARDWARE SWITCH REGISTER IF NOT FOUND OR IT IS
::EQUAL TO A -1. SET UP FOR A SOFTWARE SWITCH REGISTER.

MOV  @D0,$ERRORVECT-(SP)  SAVE ERROR VECTOR
MOV  $44, @D0,$ERRORVECT  SETUP ERROR VECTOR
MOV  @D0,HARDWAREVECTOR  SET UP FOR A HARDWARE SWITCH REGISTER
MOV  @D0,$HARDWAREVECTOR  AND A HARDWARE DISPLAY REGISTER
CMP  @D0,$DISPLAY  TRY TO REFERRENCE HARDWARE REGISTER
BNE  66$  BRANCH IF NO TIMEOUT TRAP OCCURRED
AND  @D0,$DISPLAY  AND THE HARDWARE SUR IS NOT -1

::POINTER TO SOFTWARE SW
:
MOV  @D0,$HARDWAREDISPLAY  BRANCH IF NO TIMEOUT
MOV  $65,$(SP)  SET UP FOR TRAP RETURN
RTI

::RESTORE ERROR VECTOR

::INITIALIZE THE IYT INTERRUPT HANDLER

::TYPE THE NAME OF THE PROGRAM IF FIRST PASS

INC  8@  FIRST TIME?
BNE  65$  BRANCH IF NO

::TYPE ASCIZ STRING

::GET VALUE FOR SOFTWARE SWITCH REGISTER

TST  @D0  ARE WE RUNNING UNDER XXDP/ACT?
BNE  69$  BRANCH IF YES
CMP  @D0,$SWREG  SOFTWARE SWITCH REG SELECTED?
BNE  70$  BRANCH IF NO
BTCMR  70$  GET SOFT-SWR SETTINGS
BR  70$  SET AUTO-MODE INDICATOR

::GET OVER THE ASCIZ

BR  67$  GET OVER THE ASCIZ

::ASCIZ (CRLF)//RK11 UTILITY PACKAGE//15\12//MAINDEC-11-DZRI-\E\CRLF)

BPL  18
GET VALUE FOR SOFTWARE SWITCH REGISTER

1$: TSTB STFLG
   BEO 108
   TABLY
   BEO 108
   INCB STFLG

1$: STR1: CLRBP OSPFG
   6$: TYPE 10401 000206
   BR 648
   6$: .ASCIZ (15<12<15<12>)
   GET OVER THE ASCIZ
   TYPE
   6$: .ASCIZ (15<12)<INDEX
   GET OVER THE ASCIZ

1$: TYPE 69$
   6$: .ASCIZ (15<12)<INDEX
   GET OVER THE ASCIZ
   TYPE
   6$: .ASCIZ (15<12)<INDEX
   GET OVER THE ASCIZ

1$: TYPE 71$
   6$: .ASCIZ (15<12)<INDEX
   GET OVER THE ASCIZ
   TYPE
   71$
   .ASCIZ (15<12)<INDEX
   OSCILLATING SEEK PACKAGE

1$: TYPE 73$
   6$: .ASCIZ (15<12)<INDEX
   GET OVER THE ASCIZ
   TYPE
   73$
   .ASCIZ (15<12)<INDEX
   FORMATTER SURFACE VERIFIER

1$: TYPE 75$
   6$: .ASCIZ (15<12)<INDEX
   GET OVER THE ASCIZ
   TYPE
   75$
   .ASCIZ (15<12)<INDEX
   RKOS CONTROL PANEL TEST

1$: TYPE 77$
   6$: .ASCIZ (15<12)<INDEX
   GET OVER THE ASCIZ
   TYPE
   77$
   .ASCIZ (15<12)<INDEX
   RKOS CONTROL PANEL TEST #2

1$: TYPE 79$
   6$: .ASCIZ (15<12)<INDEX
   GET OVER THE ASCIZ
   TYPE
   79$
   .ASCIZ (15<12)<INDEX
   HEAD ALIGNMENT ROUTINE

1$: TYPE 81$
   6$: .ASCIZ (15<12)<INDEX
   GET OVER THE ASCIZ
   TYPE
   81$
   .ASCIZ (15<12)<INDEX
   POWER FAILURE (WRITE) TEST

1$: 65$
   6$: .ASCIZ (15<12)<INDEX
   GET OVER THE ASCIZ
   TYPE
   65$
   .ASCIZ (15<12)<INDEX
   GET OVER THE ASCIZ
   TYPE
   65$
   .ASCIZ (15<12)<INDEX
   GET OVER THE ASCIZ

1$: RDOCT
   MOV (SP)+, RO
   STORE IT IN RO
   CMP #10, RO
   VALID NUMBER?
   BLE NG
   BR IF NOT
   RO
   ALIGN THE NUMBER FOR DISPATCHING
   TMP #BEGIN(RO)
   GO TO THE SELECTED TEST
   NG:
   TYPE 65$
   GET OVER THE ASCIZ
   TYPE
   65$
   .ASCIZ (15<12)<INDEX
   TYPE
BEGIN: STAT

SECT. 3

AUTSL: 100401 002652

SECT. 4

TYPE 65%

SECTION:

TYPE ASCIZ STRING

GET OVER THE ASCIZ

BR 64%

1065%: .ASCIZ (\15\12)/TERMINATE WITH ".\CR"/

64%

TYPE 67%

SECTION:

GET OVER THE ASCIZ

BR 66%

1067%: .ASCIZ (\15\12)/DRIVE NUMBERS ON SYSTEM 1=

66%

RDIN

MOV (SP)+, RO

GET THE ADDRESS OF THE INPUT BUFFER

MOV 1281, RO

GET THE ADDRESS OF THE LOGICAL UNIT TABLE.

CLR 1281

CLEAR THE DRIVE COUNTER

1$: CLR 1281

CLEAR TEMP

MOV (RO)+, 1281

GET THE FIRST DRIVE &

CMPB #84, 1281

IS IT A COMMA THAT WAS TYPED?

BEQ 1%

IF YES GO BACK

MAKE ASCII A DRIVE &

BML 2%

IF RESULT NEGATIVE BRANCH

SUB #60, 1281

IF RESULT POSITIVE JUMP

JSR 100401

AFTER STORING GET NEXT &

BR 1%

HAS NEGATIVE RESULT A TERMINATOR?

2$: CMPB #84, (RO)

IF YES BRANCH

JSR 100401

IF NO BAD CHARACTER JUMP

102740, 000056

IS THE TABLE FULL

BR 3%

IF YES BRANCH

MOV #000000, (R1)+

IF NO TABLE FULL WITH DOWN INDICATOR.

:ROUTINE TO DETERMINE IF THERE IS A SECOND

SYSTEM AND IF SO TO GET THE NUMBER OF THE

DRIVE ON THIS SYSTEM

:SECSYS:

TYPE 66%

SECTION:

TYPE ASCIZ STRING

GET OVER THE ASCIZ

BR 64%

1065%: .ASCIZ (\15\12) IS THERE A SECOND SYSTEM?
READ A CHARACTER
ROCHR
+ MOV
(SP)+, @KYTEMP
GET THE RESPONSE
ECHO
TYPE @KYTEMP
ECHO
CMP $131, @KYTEMP
HAS IT A "Y" (FOR YES)?
BEG
PRO2
IF YES BRANCH TO PROCESSOR 2
CMP $116, @KYTEMP
HAS RESPONSE LEGAL (N FOR NO)?
BEG
PRO1
IF LEGAL BRANCH
TYPE @ASCIIZ STRING
BR $65
GET OVER THE ASCIZ

SECSYS
BR $68
GO BACK ASK AGAIN
SET FLAG TWO PROCESSORS
NOP

PRO2
BSB $377, $PRONUM
TYPE @ASCIIZ STRING
BR $68
GET OVER THE ASCIZ

DRIVE #

BADINP:
TYPE @ASCIIZ
BR $64
GET OVER THE ASCIZ

PRO1:
CLR $PRONUM
CLEAR THE FLAG ONE PROCESSOR

GO:
MOV $LOGA, RO
BR $GO2
BRANCH AROUND INCREMENT ROUTINE

GET THE TABLE ADDRESS TO RO
CLR $INDEX
CLR $LOGA

ADD $2, RO
INDEX THRU THE TABLE
DOME

ADD $DRVO, RO
IS THE DRIVE ACTIVE
IF YES GET OUT

BCQ EXIT
IS THE DRIVE ACTIVE
IF YES BRANCH

IS THE NEXT ONE

MOV $RO, @DRACY
GET THE ACTIVE DRIVE WORD
JSR PC, CYCLE
CALL CYCLE (PICK UP DRIVE & CYL BASE, AND CALL MOUNT)
EXIT:

EXITA2:

EXITX:

; TYPE ASCIZ STRING
; GET OVER THE ASCIZ
; JMP RESTART

; THIS IS PROCESSOR #2 CODE. THIS ROUTINE ASKS FOR AND UNPACKS THE CONTROL
; WORDS FROM THE FIRST PROCESSOR.

SECOND:

; TYPE ASCIZ STRING
; GET OVER THE ASCIZ
; MOV R3, (SP)
; GET READY TO TYPE WORD COUNTER

; TYPE ASCIZ STRING
; GET OVER THE ASCIZ

; PICK UP THE OCTAL WORD
; GET THE FIRST DRIVE TO RI
; GET THE SECOND DRIVE TO RO LOW BYTE
; CLEAR THE UNUSED BITS
; CLEAR THE UNUSED BITS
; ROTATE RIGHT RD
; IF CARRY IS CLEAR BRANCH
; SET DOWN BIT IF CARRY SET
; AND CLEAR THE CARRY BIT
; MOV THE SAME FOR RI
004 02624 06701 000200
3%: BIS BIT7 R1 | IF ERROR SET THE BIT
  JSR PC, MASKER | SET DRIVE FIRST IN TABLE
  MOV R1, R1 | Call the mask control subroutine
  JSR PC, MASKER | Set drive second in table (next word
  MOV R0, R2 | Call all the mask control subroutine
  INC R3 | increment the word counter
  JC INC | go back and get next word
  INC R5 | fill the table (last word)
EXIT: BIS R4, (R2) | WITH ALL BITS SET
  ROR R4 | go back if not done
  MOV R0, (R2) | clear down and second system bits at table
  MOV R2, R0 | get address to R0 (current table)
  ST (R2)+ | ADD 2 TO THE POINTER
  BLKON: CMP 0 @DRVD, R2 | TABLE FULL
  BR EQ 29 | IF YES BRANCH
  MOV R2, R0 | fill the table
  BR FILKON | go back try again
2%: MOV R0, (R1)+ | GET THE WORD TO R1
  MOV R1, R2 | Form drive # for mount
  JP R1, 00531 | read sample (all drives)
  JSR PC, 00534 | type ASCII string
  JSR PC, 00537 | GET OVER THE ASCII
  JSR PC, 00540 | ASCII (15<12>done system 2 restart system 1, type word
  BR 00543 | 64%: MOV (R0), -(SP) | get word for system 1 and type
  TOPOS | system word, if a drive is down no bit is set but the mask
  BYTE 6 | is shifted.
1%: HALT | 63%: MASKER: TST (R2) | is the drive up
  RETRAN: R0, R2 | if no branch
  MOV R4, (R2) | move the mask bit in the table
  RETRAN: R0, R4 | rotate the mask
  BCC 15 | done; if no branch
  MOV @EXITB, (SP) | set up for return
  RTS PC | 62%: RETRAN: R0, R2 | is this system # 2's drive?
  MOV @EXITA, (SP) | if no branch
  RTS PC | set up for return
  RTS PC | index thru logical table
1%: MOV R0, R2 | 61%: ROUTINE TO BUILD THE ACTIVE LOGICAL UNIT TABLE
  RTS PC | STORE: CMP #10, #K*TEMP | is input a legal number?
BLE ILEGAL
ALIGN DRIVE # FOR TABLE
IKE TEMP
PUT THE WORD IN THE TABLE
CLR X TEMP
CLEAR THE TEMP WORD
INC X DEXCT
INCREASE THE COUNTER
CMP X DRIO,R1
IS THE TABLE FULL?
BEQ BFTUL
IF YES BRANCH
IF NO BRANCH
PC
IF TABLE FULL SET UP FOR SYSTEM #2
RETURN
TBLFUL: MOV X SECSYS,(SP)
IF ILLEGAL RESPONSE, GO BACK
RTS PC
RETURN
ILEGAL: MOV X BRTSL2,(SP)
TYPE ASCIZ STRING
BR 64%
64%: ASCIZ /ILLEGAL INPUT/
RTS PC
RETURN

THIS ROUTINE GETS A DRIVE # AND BUILDS A TABLE OF CYLINDER ADDRESS FOR USE BY WRITE (RO)=ADDRESS OF ACTIVE WORD IN LOGICAL TABLE. IT ALSO SETS AND CLEAR THE MASK BITS OF THE TABLE AS OPERATIONS INDICATE

CYCLE: MOV (RO),R1
GET THE LOGICAL UNIT ACTIVE WORD TO R1
BIT #BIT4,R1
IS IT ON SYSTEM #2
BEQ CYCLE
IF NO BRANCH
JMP #SECOME
GO TO SYSTEM #2
MOVB INDEX R3
GET THE INDEX VALUE
MOVX MSKTLB.(R3),R2
GET THE MASK TO R2
JSR PC,MASK
CALL THE MASK SUBROUTINE
MOVX (RO),R2
GET THE ADDRESS TO R2
LDLX: MOV RLCX,R1
2%:
CMP R0,R3
IS ACTIVE ADDRESS TO FIRST ADDRESS
BNE 3%
IF NO BRANCH
BIS R0,(R3)+
IF YES SET BITS TO SHOW WRITE
BR 4%
BRANCH TO SEE IF DONE
3%: BIC R2,(R3)+
CLEAR BITS TO SHOW OVERWRITE
4%:
CMP DRIO,R3
DONE
BNE 2%
IF NO GO BACK
D02: SHWB R1
GET DRIVE # TO LOW BYTE
***
NOP
GET TO R2
NOVX R0,R12
CLEAR THE UNUSED BITS
MOVX R12,XDRIVE
GET THE DRIVE #
ROL R2
SHIFT THE DRIVE # TO
ROL R2
ALIGN IT FOR THE DRIVE ADDR.
ROL R2
KEEP IT MOVING!
ROL R2
A LITTLE MORE!
ROL R2
THERE IT IS
MOVX R3,R2
RDSKTEMP+1
GET IT TO DISK ADDR. TEMP.
JSR PC,MOUNT
CALL MOUNT
RTS PC
RETURN

MOUNT: INCX #INDEX
INCREASE THE INDEX

NOP
MOV #5,#ECNT
SET ERROR CNT TO 5
; THIS ROUTINE Initializes A DRIVE AND INSURES THAT IT IS READY AND
; WRITE ENABLED, IT IS ENTERED FROM MOUNT OR FROM EXECUTE IF OPERATION FAILS

; INITIAL: MOV RD,-(SP) ; SAVE RD
; MOV -21:DSADR,RD ; GET THE DRIVE # TO RD
; CLR @TIM & ; CLEAR THE UNUSED BITS
; CLR @TIM & ; CLEAR THE TIMER
; MOV RD,DRAK OA ; GET DRIVE # TO 'DA' REGISTER
; MOV @1:DRKCS ; ISSUE CONTROL RESET + GO
; TS @SAT & ; DID CONTROL READY SET
; BMI @TIM & ; IF YES BRANCH
; BR @TIM & ; INCREMENT THE TIMER
; BR @0 & ; IF TIMER NOT ZERO BRANCH
; TYPE @5: ; TYPE ASCIZ STRING
; BR @6: ; GET OVER THE ASCIZ
; ; ASCIZ <15>(12) ; CONTROL RESET TIME OUT/
; MOV RD,DRA KOA ; DRIVE NUMBER TO 'DA' REG.
; TS @DRKO & ; DRIVE READY
; BMI @5: ; IF YES BRANCH
; TYPE @6: ; TYPE ASCIZ STRING
; BR @6: ; GET OVER THE ASCIZ
; ; ASCIZ <15>(12) ; DRIVE NOT READY/
; BR @5: ; GO BACK TRY AGAIN!
; MOV -21:BIT,DRK OA ; IS DRIVE WRITE LOCKED?
; BR @5: ; IF YES BRANCH
; TYPE @6: ; TYPE ASCIZ STRING
; BR @6: ; GET OVER THE ASCIZ
; ; ASCIZ <15>(12) ; DRIVE WRITE PROTECTED/
; BR @5: ; YES, GO BACK TRY AGAIN
; GET THE DRIVE # TO 'DA' REGISTER
MOV RO, JRDA

; ISSUE DRIVE RESET + GO
JSR PC, SRMTH

; READ/WRITE/SEEK READY BIT SET?
BIT #100, JRDS
BNE 5

; IF YES, BRANCH
BNE 6

; NO, INCREMENT THE TIMER
INC #10, #THM2

; GO BACK AND CHECK IF TIMER NOT O
BR 5

; TYPE ASCIZ STRING
Type 71s

; GET OVER THE ASCIZ
BR 70s

; ASCIZ (IS)<12> DRIVE RESET TIMED OUT/
BR 70s

; GO BACK, TRY AGAIN
BR 4s

; RESTORE RO
MOV (SP)+, RO

; RETURN TO CALLER
RTS PC

; THIS ROUTINE TAKES CARE OF ALL LINKAGE FOR THE
; EXECUTE ROUTINE, IT FORMS THE ADDRESS AND SETS UP ALL THE
; REGISTERS FOR THE WRITE OPERATION

; WRINK: CLR R1 2&CMDD
; INDICATE WRITE OPERATION
NOP

; PICK UP THE DRIVE #
MOV A, DRIVE

; MAKE IT A WORD INDEX
MOV R1, SI

; PULL THE DRIVE PATTERN
MOV #DRVAL(R1), PATT1

; CALL CYLINDER ADDRESS
JSR PC, CYLADR

; RETURN HERE IF NOT LAST BASE
BR 2s

; RETURN HERE IF LAST BASE
RTS PC

; GET THE ADDRESS OF THE OUTPUT
MOV PATT1, #USDR

; GET THE CONTROL + STATUS WORD
MOV #DRWOOD, PC

; CALL EXECUTE
JSR PC, EXECUTE

; HAS THIS WRITE SURFACE "1"?
BNE 3s

; IF YES BRANCH
BNE 1s

; SET SURFACE ONE BIT
BIS #BIT4, #DOSKAD

; SET SURFACE ONE DATA
BIS #BIT1, #DOSKAD

; RELEASE REGISTERS AND EXECUTE
RELD PC

; MAKE IT SURFACE 0 DATA
MOV #BIT1, #DOSKAD

; INC. THEN SELECTED CYL. OFFSET TABLE
INC #THRE

; GET THE CYLINDER VALUE
JSR PC, CYLOFF

; RETURN HERE IF MORE TO READ
RTS PC

; THIS ROUTINE EXPECTS TO FIND A MASK IN R5, AND FROM THIS MASK
; BUILD A TABLE (AT C Y T B L ) OF CYLINDER ADDRESS OFFSETS; THE TABLE
; IS TERMINATED BY A #377

; SAVE R5
MOV R5, -(SP)
; THIS ROUTINE FORMS A CYLINDER ADDRESS FOR BOTH THE READ AND WRITE ROUTINES
; WHEN THE BASE TABLE IS FULLY INDEXED IT RETURNS TO PC+2 OF CALLER

CYLADR: MOV #BASE, R3          ; GET THE CYLINDER TABLE ADDRESS
        MOV #CYTBL, R2          ; GET THE SELECTED CYL BASE ADDR.
        MOVB (R2), R4          ; GET THE SELECTED CYL VALUE TO R4
        NOP
        CMPB #377, R4          ; IS IT THE TABLE TERMINATOR?
        BEQ  BASINC
        CLR - (SP)             ; INSURE CLEAN WORD
        MOVB (R3), (SP)        ; GET THE CYL ADDRESS ON THE STACK
        ADD (SP)+, R4          ; AND IT TO THE SELECTED OFFSET
        ROL R4                 ; SHIFT THIS RESULT
        ROL R4                 ; TO ALIGN THE NEWLY FORMED
        ROL R4                 ; CYLINDER ADDRESS WITH Bits
        ROL R4                 ; THRU 12 OF BASE AND
        ROL R4                 ; STORE THIS IN DSKADR
        BIC #01777, R4, #DSKADR ; CLEAR ALL BUT DRIVE NUMBER
        ROL R4, #DSKADR        ; PUT IT IN DSKADR
        BASINC: INC R3         ; PICK UP ADDRESS OF NEXT BASE CYL.
        NOP
        CMP #000200            ; ARE YOU FINISHED?
        BEQ  RETRN3
        ACYTLBL, R3
        RETRN3: ADD #0274, R3
        SET-UP FOR PC+2
        RTS PC
        RET

; ROUTINE TO PERFORM INDICATED FUNCTION AND CHECK FOR
; DONE AND ERRORS

EXECUT: CLR #TMR             ; CLEAR THE TIMER
        NOP
        CLR B #TMR2
        MOV #DSKADR, RAKDA
        MOV #BUSADR, RKB
        MOV #WDCNT, RAKW
        CLR RAKCS
        CLR SMTE
        MOV #0274, R3
        BPL TIME
        RET
N03

CPU1: 24-MAR-78 09:20
CPU2: 24-MAR-78 09:23

; 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.

RDLINK: MOV RO,-(SP) ; SAVE RO

MOVL 0001360

MOVL 0001360

JST DERRFLG ; IF NO BRANCH

CLR DERRFLG ; CLEAR THE FLAG

BR Exec ; TRY AGAIN

; TIME: INC @TMR ;***

NOP

BNE CHECK1 @TMR2 ; SECOND TIMEOUT?

BNE 1 ; IF YES BRANCH

INC @TMR2 ; INDICATE SECOND TIMEOUT

BR CHECK1 ; GO BACK

; JSR PENVINITIL

; TYPE ASCII STRING

; GET OVER THE ASCII

; 165% .ASCIIZ (15<12)/TIMED OUT ON OPERATION RETRY IN PROGRESS/ 64%

BR Exec

MOV 005000, @TMR

BR Exec

MOV 001346 @TMR

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.

RDLink: MOV RO,-(SP) ; SAVE RO

MOVL 0001360

MOVL 0001360

JST DERRFLG ; IF NO BRANCH

CLR DERRFLG ; CLEAR THE FLAG

BR Exec ; TRY AGAIN

; TIME: INC @TMR ;***

NOP

BNE CHECK1 @TMR2 ; SECOND TIMEOUT?

BNE 1 ; IF YES BRANCH

INC @TMR2 ; INDICATE SECOND TIMEOUT

BR CHECK1 ; GO BACK

; JSR PENVINITIL

; TYPE ASCII STRING

; GET OVER THE ASCII

; 165% .ASCIIZ (15<12)/TIMED OUT ON OPERATION RETRY IN PROGRESS/ 64%

BR Exec

MOV 005000, @TMR

BR Exec

MOV 001346 @TMR

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.

RDLink: MOV RO,-(SP) ; SAVE RO

MOVL 0001360

MOVL 0001360

JST DERRFLG ; IF NO BRANCH

CLR DERRFLG ; CLEAR THE FLAG

BR Exec ; TRY AGAIN

; TIME: INC @TMR ;***

NOP

BNE CHECK1 @TMR2 ; SECOND TIMEOUT?

BNE 1 ; IF YES BRANCH

INC @TMR2 ; INDICATE SECOND TIMEOUT

BR CHECK1 ; GO BACK

; JSR PENVINITIL

; TYPE ASCII STRING

; GET OVER THE ASCII

; 165% .ASCIIZ (15<12)/TIMED OUT ON OPERATION RETRY IN PROGRESS/ 64%

BR Exec

MOV 005000, @TMR

BR Exec

MOV 001346 @TMR

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.
804

COMPARABILITY TEST

EXIT2: MOV (SP)+,RO

THE ERROR CHECK ROUTINE CHECKS IF AN ERROR OCCURRED

; ON WRITING OR READING.

ERCHK: BIT #140000, DRKCS

HARD ERROR OR ERROR SET?

; WAIT HERE TO EXAMINE ERROR REG., ECT.

ERR: MOV TSTSN1

IF NO, GO TEST 'SIN' BIT

ERR1: MOV #1, ERRFLG

IF YES, ISSUE CTRL RESET + GO

ERR2: MOV #1, ERRFLG

FLAG AN ERROR (PREVENT UPDATE OF ADDR)

ERR3: MOV #1, ERRFLG

CTRL READY BIT SET (FROM CTRL RESET)

ERR4: MOV #0, ERRFLG

IF NO WAIT! (IF HUNG HERE RUN STATIC)

ERR5: MOV #0, ERRFLG

DECEDENT ERROR COUNTER

ERR6: MOV #0, ERRFLG

HAVE ERROR BITS SET 5 TIMES?

ERR7: MOV #0, ERRFLG

SEEK INCOMPLETE SET?

ERR8: MOV #0, ERRFLG

BANCH IF NO

ERR9: MOV #0, ERRFLG

IF YES, ISSUE DRIVE RESET, GO

ERR10: MOV #0, ERRFLG

FLAG AN ERROR (PREVENT UPDATE OF ADDR)

; "RDY READY" BIT SET?

ERR11: MOV #0, ERRFLG

IF NO WAIT! (IF HUNG HERE RUN STATIC)

ERR12: MOV #0, ERRFLG

DECEDENT SEEK INCOMPLETE COUNTER

ERR13: MOV #0, ERRFLG

IF '3 'SIN' ERRORS FALL THROUGH

ERR14: MOV #0, ERRFLG

; TYPE ASCI STRING

ERR15: MOV #0, ERRFLG

GET OVER THE ASCI

ERR16: MOV #0, ERRFLG

; "RDY READY" BIT SET?

ERR17: MOV #0, ERRFLG

IF NO WAIT! (IF HUNG HERE RUN STATIC)

ERR18: MOV #0, ERRFLG

DECEDENT SEEK INCOMPLETE COUNTER

ERR19: MOV #0, ERRFLG

IF '3 'SIN' ERRORS FALL THROUGH

ERR20: MOV #0, ERRFLG

; TYPE ASCI STRING

ERR21: MOV #0, ERRFLG

GET OVER THE ASCI

ERR22: MOV #0, ERRFLG

; "RDY READY" BIT SET?

ERR23: MOV #0, ERRFLG

IF NO WAIT! (IF HUNG HERE RUN STATIC)

ERR24: MOV #0, ERRFLG

DECEDENT SEEK INCOMPLETE COUNTER

ERR25: MOV #0, ERRFLG

IF '3 'SIN' ERRORS FALL THROUGH

ERR26: MOV #0, ERRFLG

; TYPE ASCI STRING

ERR27: MOV #0, ERRFLG

GET OVER THE ASCI

ERR28: MOV #0, ERRFLG

; "RDY READY" BIT SET?

ERR29: MOV #0, ERRFLG

IF NO WAIT! (IF HUNG HERE RUN STATIC)

ERR30: MOV #0, ERRFLG

DECEDENT SEEK INCOMPLETE COUNTER

ERR31: MOV #0, ERRFLG

IF '3 'SIN' ERRORS FALL THROUGH

ERR32: MOV #0, ERRFLG

; TYPE ASCI STRING

ERR33: MOV #0, ERRFLG

GET OVER THE ASCI

ERR34: MOV #0, ERRFLG

; "RDY READY" BIT SET?

ERR35: MOV #0, ERRFLG

IF NO WAIT! (IF HUNG HERE RUN STATIC)

ERR36: MOV #0, ERRFLG

DECEDENT SEEK INCOMPLETE COUNTER

ERR37: MOV #0, ERRFLG

IF '3 'SIN' ERRORS FALL THROUGH

ERR38: MOV #0, ERRFLG

; TYPE ASCI STRING

ERR39: MOV #0, ERRFLG

GET OVER THE ASCI

ERR40: MOV #0, ERRFLG

; "RDY READY" BIT SET?

ERR41: MOV #0, ERRFLG

IF NO WAIT! (IF HUNG HERE RUN STATIC)

ERR42: MOV #0, ERRFLG

DECEDENT SEEK INCOMPLETE COUNTER

ERR43: MOV #0, ERRFLG

IF '3 'SIN' ERRORS FALL THROUGH

ERR44: MOV #0, ERRFLG

; TYPE ASCI STRING

ERR45: MOV #0, ERRFLG

GET OVER THE ASCI

ERR46: MOV #0, ERRFLG

; "RDY READY" BIT SET?

ERR47: MOV #0, ERRFLG

IF NO WAIT! (IF HUNG HERE RUN STATIC)

ERR48: MOV #0, ERRFLG

DECEDENT SEEK INCOMPLETE COUNTER

ERR49: MOV #0, ERRFLG

IF '3 'SIN' ERRORS FALL THROUGH

ERR50: MOV #0, ERRFLG

; TYPE ASCI STRING

ERR51: MOV #0, ERRFLG

GET OVER THE ASCI

ERR52: MOV #0, ERRFLG

; "RDY READY" BIT SET?

ERR53: MOV #0, ERRFLG

IF NO WAIT! (IF HUNG HERE RUN STATIC)

ERR54: MOV #0, ERRFLG

DECEDENT SEEK INCOMPLETE COUNTER

ERR55: MOV #0, ERRFLG

IF '3 'SIN' ERRORS FALL THROUGH

ERR56: MOV #0, ERRFLG

; TYPE ASCI STRING

ERR57: MOV #0, ERRFLG

GET OVER THE ASCI

ERR58: MOV #0, ERRFLG

; "RDY READY" BIT SET?

ERR59: MOV #0, ERRFLG

IF NO WAIT! (IF HUNG HERE RUN STATIC)

ERR60: MOV #0, ERRFLG

DECEDENT SEEK INCOMPLETE COUNTER

ERR61: MOV #0, ERRFLG

IF '3 'SIN' ERRORS FALL THROUGH

ERR62: MOV #0, ERRFLG

; TYPE ASCI STRING

ERR63: MOV #0, ERRFLG

GET OVER THE ASCI

ERR64: MOV #0, ERRFLG

; "RDY READY" BIT SET?
; THIS ROUTINE CHECKS A SECTOR'S WORTH OF DATA ON A READ OPERATION
; IT ALLOWS 8 ERROR PRINTOUTS PER SECTOR

ROCHK: MOV RN,-(SP) ; SAVE RN
         MOV RS,-(SP) ; SAVE RS FOR RDLINK
         CLRB $$HDFLG ; CLEAR THE PRINT HEADER FLAG
         NOP
         MOV $$ $$HCCHKCINT ; PUT ERROR COUNT IN CHECK COUNT
         MOV $$ $$HDFLG ; GET THE TABLE ADDRESS TO RN
         MOV $$ $$HDFLGS, RS ; GET THE EXPECTED DATA TO RS
         CMP RS, $$RN+ ; ARE THEY THE SAME
         BNE 3$ ; IF NOT BRANCH
         MOV $$ $$HDFLGS, RN ; IF NO BRANCH
         JMP $$ $$HDFLGS ; THE HEADER FLAG CLEAR
         BNE 2$ ; IF NO BRANCH
         TYPE 6$ ; TYPE ASCII STRING
         BR 6$ ; GET OVER THE ASCII
         .65$; ASCII $<15><12>/ERROR! DATA WRITTEN BY DRIVE /
         .64$; ASCII $<15><12>/ERROR! DATA WRITTEN BY DRIVE /
         .63$; ASCII $<15><12>/ERROR! DATA WRITTEN BY DRIVE /
         .62$; ASCII $<15><12>/ERROR! DATA WRITTEN BY DRIVE /
         .61$; ASCII $<15><12>/ERROR! DATA WRITTEN BY DRIVE /
         .60$; ASCII $<15><12>/ERROR! DATA WRITTEN BY DRIVE /
         .59$; ASCII $<15><12>/ERROR! DATA WRITTEN BY DRIVE /
         .58$; ASCII $<15><12>/ERROR! DATA WRITTEN BY DRIVE /
         .57$; ASCII $<15><12>/ERROR! DATA WRITTEN BY DRIVE /
         .56$; ASCII $<15><12>/ERROR! DATA WRITTEN BY DRIVE /
         .55$; ASCII $<15><12>/ERROR! DATA WRITTEN BY DRIVE /
         .54$; ASCII $<15><12>/ERROR! DATA WRITTEN BY DRIVE /
         .53$; ASCII $<15><12>/ERROR! DATA WRITTEN BY DRIVE /
         .52$; ASCII $<15><12>/ERROR! DATA WRITTEN BY DRIVE /
         .51$; ASCII $<15><12>/ERROR! DATA WRITTEN BY DRIVE /
         .50$; ASCII $<15><12>/ERROR! DATA WRITTEN BY DRIVE /
         .49$; ASCII $<15><12>/ERROR! DATA WRITTEN BY DRIVE /
         .48$; ASCII $<15><12>/ERROR! DATA WRITTEN BY DRIVE /
         .47$; ASCII $<15><12>/ERROR! DATA WRITTEN BY DRIVE /
         .46$; ASCII $<15><12>/ERROR! DATA WRITTEN BY DRIVE /
         .45$; ASCII $<15><12>/ERROR! DATA WRITTEN BY DRIVE /
         .44$; ASCII $<15><12>/ERROR! DATA WRITTEN BY DRIVE /
         .43$; ASCII $<15><12>/ERROR! DATA WRITTEN BY DRIVE /
         .42$; ASCII $<15><12>/ERROR! DATA WRITTEN BY DRIVE /
         .41$; ASCII $<15><12>/ERROR! DATA WRITTEN BY DRIVE /
         .40$; ASCII $<15><12>/ERROR! DATA WRITTEN BY DRIVE /
         .39$; ASCII $<15><12>/ERROR! DATA WRITTEN BY DRIVE /
         .38$; ASCII $<15><12>/ERROR! DATA WRITTEN BY DRIVE /
         .37$; ASCII $<15><12>/ERROR! DATA WRITTEN BY DRIVE /
         .36$; ASCII $<15><12>/ERROR! DATA WRITTEN BY DRIVE /
         .35$; ASCII $<15><12>/ERROR! DATA WRITTEN BY DRIVE /
         .34$; ASCII $<15><12>/ERROR! DATA WRITTEN BY DRIVE /
         .33$; ASCII $<15><12>/ERROR! DATA WRITTEN BY DRIVE /
         .32$; ASCII $<15><12>/ERROR! DATA WRITTEN BY DRIVE /
         .31$; ASCII $<15><12>/ERROR! DATA WRITTEN BY DRIVE /
         .30$; ASCII $<15><12>/ERROR! DATA WRITTEN BY DRIVE /
         .29$; ASCII $<15><12>/ERROR! DATA WRITTEN BY DRIVE /
         .28$; ASCII $<15><12>/ERROR! DATA WRITTEN BY DRIVE /
         .27$; ASCII $<15><12>/ERROR! DATA WRITTEN BY DRIVE /
         .26$; ASCII $<15><12>/ERROR! DATA WRITTEN BY DRIVE /
         .25$; ASCII $<15><12>/ERROR! DATA WRITTEN BY DRIVE /
         .24$; ASCII $<15><12>/ERROR! DATA WRITTEN BY DRIVE /
         .23$; ASCII $<15><12>/ERROR! DATA WRITTEN BY DRIVE /
         .22$; ASCII $<15><12>/ERROR! DATA WRITTEN BY DRIVE /
         .21$; ASCII $<15><12>/ERROR! DATA WRITTEN BY DRIVE /
         .20$; ASCII $<15><12>/ERROR! DATA WRITTEN BY DRIVE /
         .19$; ASCII $<15><12>/ERROR! DATA WRITTEN BY DRIVE /
         .18$; ASCII $<15><12>/ERROR! DATA WRITTEN BY DRIVE /
         .17$; ASCII $<15><12>/ERROR! DATA WRITTEN BY DRIVE /
         .16$; ASCII $<15><12>/ERROR! DATA WRITTEN BY DRIVE /
         .15$; ASCII $<15><12>/ERROR! DATA WRITTEN BY DRIVE /
         .14$; ASCII $<15><12>/ERROR! DATA WRITTEN BY DRIVE /
         .13$; ASCII $<15><12>/ERROR! DATA WRITTEN BY DRIVE /
         .12$; ASCII $<15><12>/ERROR! DATA WRITTEN BY DRIVE /
         .11$; ASCII $<15><12>/ERROR! DATA WRITTEN BY DRIVE /
         .10$; ASCII $<15><12>/ERROR! DATA WRITTEN BY DRIVE /
         .09$; ASCII $<15><12>/ERROR! DATA WRITTEN BY DRIVE /
         .08$; ASCII $<15><12>/ERROR! DATA WRITTEN BY DRIVE /
         .07$; ASCII $<15><12>/ERROR! DATA WRITTEN BY DRIVE /
         .06$; ASCII $<15><12>/ERROR! DATA WRITTEN BY DRIVE /
         .05$; ASCII $<15><12>/ERROR! DATA WRITTEN BY DRIVE /
         .04$; ASCII $<15><12>/ERROR! DATA WRITTEN BY DRIVE /
         .03$; ASCII $<15><12>/ERROR! DATA WRITTEN BY DRIVE /
         .02$; ASCII $<15><12>/ERROR! DATA WRITTEN BY DRIVE /
         .01$; ASCII $<15><12>/ERROR! DATA WRITTEN BY DRIVE /
         .00$; ASCII $<15><12>/ERROR! DATA WRITTEN BY DRIVE /
D04

; COMPATIBILITY TEST

; THIS ROUTINE BUILDS A TABLE OF PARAMETERS TO PASS TO THE SECOND SYSTEM
; IT PACKS THE INFO FOR THE SECOND DRIVE INTO ONE WORD

SECON. CLR R3 ; CLEAR THE WORD COUNTER

; CLEAR THE WORD COUNTER

MOV @MSKB1B,R2

; MOV @MSKB1B,R2

CMP @PRSB1B,R2

; CMP @PRSB1B,R2

BNE 65

; BNE 65

MOV 48024, R0

; MOV 48024, R0

; GET THE ACTIVE TABLE ADDRESS

1% MOV (R0)+, R1

; PICK UP THE WORD

SUBB R1

; GET THE DRIVE # TO THE LOW BYTE

MOV 177400, R1

; CLEAR THE UNWANTED BITS

ROR R1

; ROTATE THE BYTE, WAS DOWN SET?

RET

; IF NO BRANCH

SETB R1

; RESTORE THE TABLE POINTER

BRA 16

; GET THE NEXT WORD FROM ACTIVE TABLE

3% MOV R1, (R2)+

; GET THE LAST DRIVE TO THE PASS TABLE

MOV @PRSB1B,R2

; RESTORE THE TABLE POINTER

TYPE 65

; TYPE ASCIZ STRING

BR 65

; GET OVER THE ASCIZ

1:65% .ASCIZ (15)<(12)>/LOAD AND START ADDRESS 210 ON SYSTEM #2/

1:64% TYPE 67

; TYPE ASCIZ STRING

BR 65

; GET OVER THE ASCIZ

1:67% .ASCIZ (15)<(12)>/AND TYPE THE BELOW WHEN ASKED FOR IT. /

1:66% INC R3

; INCREMENT THE WORD COUNTER

TYPE 69

; TYPE ASCIZ STRING

BR 68

; GET OVER THE ASCIZ

1:68%
MOV R3, -(SP) ; GET THE WORD COUNT ON THE STACK

TYPE 6
BYTE 0
BYTE 00

TYPE ASCIZ 

GET OVER THE ASCIZ

GET THE FIRST TO THE STACK

HAS THIS THE TABLE TERMINATOR
BRANCH IF YES
TERMINATOR?
IF NO BRANCH

TYPE ASCIZ 

GET OVER THE ASCIZ

GET THE WORD FROM SYSTEM 2 TO TABLE
GET POINTER LOOK FOR FIRST "JP" DRIVE
DRIVE UP?
IF NO BRANCH

CALL DD2+
CALL READ CHECK
GO TO END OF TEST

TABLE TERMINATOR

SBTTL OSCILLATING SEEK ROUTINE
; DISK I/O SUBROUTINE.

; SET UP FOR A WRITE/FORMAT.
10: MOV DC, CPCX ; SET UP THE WORD COUNT REG.
    MOV DA, 3H ; SET UP THE DISK ADDRESS.
    BIS #SKT, CPCX ; SET THE UNIT NUMBER UP.
    MOV XA, XA ; SET THE BUSS ADDRESS.
    MOV #0, CPCX ; CLEAR THE CONTROL REG. FOR SET UP.
    BIS #BIT10+BIT11, CPCX ; SET FORMAT&INHIBIT INC. BIT.
    MOV B, #16 ; SET UP WRITE FUN.
    MOV #0, #120016 ; SECTOR PACK.
    MOV BI, #BIT0, CPCX ; GO TO THE WRITE FORMAT.
    MOV #0, ERR ; IS WRITE FORMAT DONE?
11: BPL #12 ; NO SO WAIT.
    BST CPCX ; WAS THERE A ERROR?
    BMI WERR ; YES GO SERVICE IT.
    MOV #0, ERR ; CLEAR OUT THE ERROR COUNTER.

; SET UP FOR A READ/FORMAT.
12: MOV RC, CPCX ; SET UP WORD COUNT REG.
    MOV DA, 3H ; SET UP DISK ADDRESS.
    BIS #SKT, CPCX ; SET THE UNIT NUMBER.
    MOV #0, #120016 ; SECTOR PACK.
    MOV BI, #BIT10, CPCX ; SET UP READ FUN.
    MOV B, #0 ; SET UP 12 OR 16 SECTOR PACK.
    MOV #0, #164432 ; SET UP THE READ FORMAT.
    MOV #0, #12712 ; SET UP 12 OR 16 SECTOR PACK.
    MOV B, #114 ; SET INHIBIT INC. BIT, STOP ON SOFT ERROR BITS.
    MOV BI, #BIT10, CPCX ; SET UP 12 OR 16 SECTOR PACK.
    MOV #0, ERR ; CLEAR OUT THE ERROR COUNTER.

; SET UP FOR A WRITE CHECK.
13: MOV DC, CPCX ; SET UP WORD COUNT REG.
    MOV DA, 3H ; SET UP DISK ADDRESS.
    BIS #SKT, CPCX ; SET THE UNIT NUMBER.
    MOV XA, XA ; SET THE BUSS ADDRESS.
    MOV #0, CPCX ; CLEAR THE CONTROL REG.
    BIS #BIT1+BIT0, CPCX ; SET INHIBIT INC. BIT, STOP ON SOFT ERROR BITS.
    MOV BI, #BIT0, CPCX ; GO TO THE WRITE CHECK.

; CHECK HEADERS READ BY THE READ/FORMAT.
; MOV RWC, R3 ; PUT NUMBER OF WORDS TO CHECK IN REG 3.
; MOV R3, R3 ; CHECK IN REG 3.
; ADD R04, R3 ; SET REG 3 TO THE LAST WORD TO BE CHECKED.
; MOV R01, R2 ; SET REG 2 TO STARTING Addr. OF BUFF.
; MORE: CMP DI, R4 ; CHECK THAT HEADER IS RIGHT.
; BEQ R4, ERRR ; THIS HEADER WAS WRONG GO SERVICE IT.
; CMP R3, R2 ; ARE WE DONE?
; BNE MORE ; NO GO CHECK THE NEXT ONE.
; MOV #0, ERRRRC ; CLEAR OUT THE ERROR COUNT.

; LETS CHECK ON THE WRITE CHECK WE STARTED.
; IS: TSTB DA1KCS ; THE CONTROLLER IS STILL BUSY.
; BPL IS ; WAS THERE A ERROR?
; BNE WCERR ; YES GO SERVICE IT.
; MOV #0, ERRRRC ; CLEAR OUT THE.
; RTS R1 ; ERROR COUNTERS.
; WCERR: JMP WCERR ; RETURN TO THE MAIN LINE.

; ERRORS FOR WRITE FORMAT.
; ERRRRC: INC ERRRRC ; ADD ONE TO THE ERROR COUNT.
; CMP #4, ERRRRC ; HAS IT HAPPENED 4 TIMES ON THIS CYL.
; BNE RETRY ; NO.
; SYSER: TYPE 6S$ ; TYPE ASCIZ STRING
; BR 64S$ ; GET OVER THE ASCIZ
; 64S$: ASCIZ <(IS)(12)>/SYSTEM ERROR/
; 64$: ; HALT START ; LET THE TECH. BREATHE.
; RETRY: MOV #0, DA1KCS ; RESTART THE TEST.
; MOV #0, DA1KCS ; CLEAR OUT THE CONTROL REG.
; MOV #0, DA1KCS ; DO A DRIVE RESET.
; IS: BIT #116, DA1KCS ; IS IT DONE?
; BNE RETRY ; NO SO WAIT.
; JMP 10 ; TRY AGAIN.

; ERRORS FOR READ/FORMAT.
; ERRRRC: INC ERRRRC ; ADD ONE TO ERROR COUNT.
; CMP #4, ERRRRC ; HAS IT HAPPENED 4 TIMES ON THIS CYL?
; BNE RETRY ; NO DO IT AGAIN.
; BR SYSER ; YES SO TELL HIM SO.

; READ/FORMAT ERRORS FOUND BY SOFTWARE CHECKS.
; ERRRRC: INC ERRRRC ; ADD ONE TO ERROR COUNT.
; CMP #4, ERRRRC ; WAIT FOR THE WRITE CHECK.
; IS: TSTB DA1KCS ; IS IT?
; BPL IS ; PUT OUT FAILED MESSAGE.
; PUT WHICH SECTORS HEADER
; FAILED IN RKDA FOR THE MESSAGE.

; TYPE 65%
; TYPE ASCIZ STRING
; GET OVER THE ASCIZ
; ASCIZ (15<12>/PACK FAILED AT (IN OCTAL) /

; GENERATE THE CYL,SECTOR,SURFACE
; MESSAGE FROM RKDA

; TYPE 18%
; TYPE OUT THE GENERATED MESSAGE
; BYPASS THE INLINE MESSAGE

; ASCII (15<12>/CYL. /

; ASCII /000 /
; ASCII /000/
; ASCII /SURF. /

; ASCIZ /0<15<12>

; STOP: HALT
; JMP START
; LET OPER DO HIS THING.
; RESTART THE TEST.

; SHIFT SUBROUTINES.
; HERE FOR A SHIFT OF 3.
; HERE FOR A SHIFT OF 2.
; HERE FOR A SHIFT OF 1.
; PUT RESULTS IN THE WORKING REG.
70%: MOV (R4)+,-(SP).
2%: TYPON #MASEL,R4 :FINISHED ALL CHECKS
4%: CMP #MASEL,R4 :IF NO BRANCH
1%: BNE 1% ,RETURN

;THE FOLLOWING REVISION WAS MADE BY JIM KAPADIA

; SBRTL CONTROL PANEL TEST #2

; THIS IS THE ENTRY POINT INTO CONTROL PANEL TEST #2. ALL
; THE DRIVES THAT ARE PRESENT AND IN 'ROY' CONDITION ARE
; REPORTED ON LINE.

; SECT. 4: NOP

1836 017300 000240 162060
1837 017300 012777 000001
1838 017310 105777 162054
1839 017314 100375
1840 017316 012700 020614
1841 017322 000501
1842 017324 000500
1843 1844 017326 010210
1845 017330 012777 162042
1846 017334 105777 162024
1847 017340 100021
1848 017342 104040 020450
1849 017346 010146
1850 017350 104042
1851 017352 104040 020461
1852 017356 052710 000300
1853 1854 017362 012777 000015 162000
1855 017370 004737 005614
1856 017374 032777 000100 161762 45% 017378
1857 017382 001774
1858 1859 017390 005720
1860 017394 005261
1861 017406 020000
1862 017414 001344
1863 017416 104401 001161
1864 1865 017422 012700 020614
1866 017426 005001
1867 017430 011077 161742
1868 017434 042777 017727 161734
1869 017442 105777 161716
1870 017446 100044
1871 017450 105710
1872 017454 0.452 100454

; 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 WAS CHANGED FROM LAST TIME THEN
; IT IS REPORTED. IF THERE IS NO CHANGE NOTHING IS REPORTED.

; BEGCT: MOV 017378

1880 017422 012700 020614
1881 017426 005001
1882 017430 011077 161742
1883 017434 042777 017727 161734
1884 017442 105777 161716
1885 017446 100044
1886 017450 105710
1887 017454 0.452 100454
GOS

1932 017460 104401 020450 TYPE EM1 ; LINE, REPORT MESSAGE
1933 017464 010146 MOV RL1,(SP)
1934 017468 104400 TYPDCE
1935 017472 020351 TYPE EM2 ; TYPE 'ON LINE'
1936 017476 000000 161662 BIT &PS,DRKDS ; WRITE ENABLED?
1937 017507 001017 BEQ 25 ; YES, OR
1938 017501 104401 CONTROL PANEL TEST #2
1939 017512 017512 TYPE EM1 ; TYPE ASCII STRING
1940 017516 000014 BR 61 ; ; GET OVER THE ASCII
1941 017543 017543 ; .ASCIIZ (ISX(12)/ERROR, NOT \n1942 017547 017277 000017 161620 WRITE PROT THE DISK
1943 017550 105777 161614 28 ; MOV #17, DRKDS
1944 017554 103777 161643 TSTB DRKCS
1945 017558 000012 BL 31
1946 017562 105710 TXT1
1947 017566 105710 11 ; TSTB (RD) ; WAS THIS DRIVE OFF LINE LAST
1948 017570 100010 BPL NXT1 ; TIME? BRANCH IF YES
1949 017574 104401 TYPE EM1 ; IF NOT REPORT THE CHANGE
1950 017578 101014 MOV RL1,(SP) ; TYPE DRIVE #
1951 017582 104402 TYPQ
1952 017586 104402 TYPE EM3 ; TYPE 'OFF LINE'
1953 017590 020473 BIC #BIT7,(RD) ; CLEAR BIT TO INDICATE THIS
1954 017594 104271 DRIVE 'OFF LINE'
1955 017598 000200
1956 017600 010577 161554 ; THIS CODE CHECKS 'WPS' BIT FOR EVERY DRIVE THAT IS IN 'DRY'
1957 017604 105777 161554 ; CONDITION (ON LINE). IT REPORTS ANY CHANGE IN THE CONDITION OF
1958 017608 100033 THE 'WPS' BIT. IF THERE WAS NO CHANGE FROM LAST TIME NOTHING
1959 017612 032777 000040 161544 IS REPORTED. AT THE TIME OF ENTRY RD POINTS TO DRIVE FLAG.
1960 017616 100014 39
1961 017620 100014 NXT1: TSTB DRKDS ; IS THIS DRIVE PRESENT?
1962 017624 100014 19 ; DRKDS CONTAINS THE DRV #
1963 017628 001043 BPL NXT2 ; NO, SKIP CHECKING
1964 017632 032777 000040 161544
1965 017636 001242 BIT #WPS,DRKDS ; WPS BIT SET?
1966 017640 001242 19 ; YES, WPS BIT CLEAR
1967 017644 032710 000004 BNE 19 ; WPS BIT CLEAR
1968 017648 032710 000004 19 ; WAS IT CLR LAST TIME ALSO?
1969 017652 010049 BEQ NXT2 ; YES, NOTHING TO REPORT.
1970 017656 104401 020450 ; WPS CHANGED FROM 'SET' TO 'CLR'. REPORT IT?
1971 017660 104401 TYPE EM1
1972 017664 101014 MOV RL1,(SP)
1973 017668 104402 ; TYPE DRIVE #
1974 017672 104402 19 TYPE EM3 ; INDICATE THAT 'WPS' IS CLEAR
1975 017676 104402 BIC #BIT2,(RD)
1976 017680 104402 19 TYPE EM5 ; TYPE 'WPS CLEAR'
1977 017684 104402 BR NXT2
1978 017688 032710 000004 19 ; WPS BIT IS SET
1979 017692 032710 000004 19 ; WAS IT SET LAST TIME ALSO?
1980 017696 001010 BNE NXT2 ; YES, WPS BIT CLEAR
1981 017699 032710 000004 19 ; WAS CHANGED FROM 'CLR' TO 'SET'. REPORT THIS CHANGE
1982 017703 104401 020450 TYPE EM1
1983 017707 101014 MOV RL1,(SP)
1984 017711 104402 ; TYPE DRIVE #
1985 017715 104402 19 TYPE EM4 ; TYPE 'WPS SET'
1986 017719 10474 052710 000004 BIS #BIT2,(RD) ; SET FLAG BIT INDICATING WPS SET
H05

This code performs a seek function on a drive and checks if the drive is set as a result. If the power has cut off from the drive, note that only those drives are checked which were found to be present at the beginning. When this test was entered, seek is done to cylinder 1 at the time of entry RD points to the drive flag.

017700 032710 000100

HXT2: BIT #BIT6,(RD) was this drive present at begng
BEQ 4:

017704 001403

NO

017708 105777 161452

TSTB #AKOS is it present now?

017712 100402 3F:

BHI 3:

017716 000137 020352

JMP DNOVR if not skip this check

4:

017720 052777 000010 161450

BIS #40,AKOA AKOA already has the DRV

3:

017724 105777 161434

MOV #11,AKCS SEEK, GO

017728 100375

1:

017732 105777 161430

TSTB #AKCS wait for control RDY

017736 100375

1:

017740 100375

BPL 1:

017744 105777 161430

MOV #40,AKOA something wrong if CNTL RDY does not come back

1:

017748 100375

BIT #DPL,AKCS DPL bit set?

1:

017752 032710 000010

BIT #BIT0,(RD) yes, DPL set

017756 001167

BEQ 2:

017760 104401 C'0450

BNE CLRDL TYPE 'EMI'

TYPE 'EMI' time, report it

017764 010146

MOV #11-(SP)

017768 104402

TYPE 'EMI'

017772 020537

BIS #BIT0,(RD) set flag bit indicating that DPL set this time

017776 052710 000001

BR CLRDL 'DPL' bit is clear

017780 000556

2:

017784 032710 000001

BIT #BIT0,(RD) was 'DPL clear last time also'

017788 104401 20450

BEQ WATSK yes, nothing to report

017792 00006 010146

TYPE 'EMI'

017796 104402

MOV #11-(SP)

017780 020560

TYPE 'EMI'

017794 104401

BIC #BIT0,(RD) type 'POWER UP'

017800 02024 042710 000001

DPL bit is set indicating that DPL is clear this time

017804 020002 032710 000001

2:

017808 104401 20450

BEQ WATSK

017812 020014 010146

TYPE 'EMI'

017800 104402

TYPE 'EMI'

017808 020020 042710 000001

DPL bit is set indicating that DPL is clear this time

017812 020030 012705 164220

WATSK:

MOV #-6000,RS

; set count to wait for
IO5

MINDEC-11-ZRKH-6
MACII 30A(1652) 24-MAR-78 09:23 PAGE 40

CONTROL PANEL TEST # 2

1$: BIT WKS, JRKDS : R/W'S. ROY SET
     BNE 3% : YES
     INC R6 : WAIT
     BNE 1% : 500 MS OVER, R/W'S RDY
     JMP 1% : DIDN'T SET. WAIT FOR
              SINT TO SET.

2$: CLR R4
     MOV #17777, R5 : SET UP COUNT
     BIT #5, JRKDS : SIN SET?
     BNE 5% : YES
     DEC R6
     BNE R4
     TST R4
     BNE 4%
     INC R4
     BR 2%
     1500 MS ELAPSED, BUT SIN
     DIDN'T SET. ERROR!

4%$: TYPE .65$
     ;TYPE ASCII STRING
     BR 64% ; GET OVER THE ASCII

64%$: MOV R1, -(SP) ;TYPE DRIVE
     ;TYPE DRIVE #

3%$: BIT #BIT1, (RD)
     BIC #BIT1, (RD)
     CLR FLAG INDICATING THAT SEEK IS OK
     MOV R1, -(SP)

5%$: TYPE JRKDS
     BR JRKDS

7%$: IF SIN SET, DO DRIVE RESET AND CLEAR IT

SIN1$: BIT #BIT1, (RD)
     BNE 4$ ; NO POST, NOTHING TO REPORT
     BIS #BIT1, (RD)
     SET FLAG INDICATING THAT
     SIN' SET, AND REPORT THE CHANGE
     MOV R1, -(SP)

9%$: TYPE , JRKDS
     ;TYPE 'SIN'

11%$: MOV JRKDS, R5
     SAVE JRKDS
     MOV #1, JRKCS
     DO CONTROL RESET
     BPL 1%
     MOV R5, JRKDA
     MOV #15, JRKCS
     WAIT FOR CONTROL ROY

13%$: MOV #105777, R5
     MOV #161140, R5
     MOV R5, JRKDA
     DO DRIVE RESET, JRKDA
     ALREADY HAS THE DRIVE

15%$: MOV JRKCS
     WAIT FOR Control ROY

17%$: MOV R105777, R5
     MOV R161104, R5
     TSTB JRKCS
     ALREADY HAS THE DRIVE

19%$: BPL 2%
DRIVE FLAGS FOR CONTROL PANEL TEST #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 BEGINNING. UNLESS 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 5 IS SET IF 'DRY', BIT GETS SET IF: DRIVE POWER OFF

BIT 4 IS SET WHEN 'DRY' IS ON, 'DRY' IS CLEARED WHEN DRIVE POWER IS ON.

BIT 3 IS SET WHEN SEEK INCOMPLETE 'SIN' OCCURS.

BIT 2 IS CLEAR WHEN SEEK IS OK

BIT 1 IS CLEAR WHEN 'DRY' IS SET FROM CONSOLE.

BIT 0 IS CLEARED WHEN DRIVE IS WRITE ENABLED.

;DRIVE FLAGS

DRIV0: .WORD 0
DRIV1: .WORD 0
DRIV2: .WORD 0
DRIV3: .WORD 0
DRIV4: .WORD 0
DRIV5: .WORD 0
DRIV6: .WORD 0
DRIV7: .WORD 0
HEAD ALIGNMENT ROUTINE

This maintenance routine is helpful in head alignment. Upon entry, the question "Drive?" is asked. The user should reply with the drive number to be aligned. If the drive is an RK-DFS, the letter 'F' is added as a suffix, for selecting surface 0. If the drive is an RK-DFS, cylinder 12 becomes cylinder 130 of the even drive, and cylinder 12 becomes cylinder 5 of the odd drive. The heads are placed on the selected cylinder and data is read continuously from the cylinder (sector 0). The upper or lower head and cylinder can be selected dynamically. If 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?).

Sect.5: NOP

<table>
<thead>
<tr>
<th>Type</th>
<th>Hex</th>
<th>ASCII String</th>
</tr>
</thead>
<tbody>
<tr>
<td>65%</td>
<td></td>
<td>GET OVER THE ASCIZ</td>
</tr>
<tr>
<td>64%</td>
<td></td>
<td>TYPE ASCII STRING</td>
</tr>
<tr>
<td>63%</td>
<td></td>
<td>15:12/SET SW0=0 FOR SURFACE 0, SW0=1 FOR SUR 1.</td>
</tr>
<tr>
<td>62%</td>
<td></td>
<td>TYPE ASCII STRING</td>
</tr>
<tr>
<td>61%</td>
<td></td>
<td>SET SW0=0 FOR CYLINDER 105, SW1=1 FOR CYLINDER 64</td>
</tr>
<tr>
<td>60%</td>
<td></td>
<td>TYPE ASCII STRING</td>
</tr>
<tr>
<td>59%</td>
<td></td>
<td>GET OVER THE ASCIZ</td>
</tr>
<tr>
<td>58%</td>
<td></td>
<td>ASCII 15:12/PUT ANY SW FROM 2-15 HI TO SELECT NEW DRIVE</td>
</tr>
</tbody>
</table>

HOALGN: TYPE EM10

<table>
<thead>
<tr>
<th>Label</th>
<th>ASCII</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>FFLAG</td>
</tr>
<tr>
<td>RDLIN</td>
<td>GET OP INPUT</td>
</tr>
<tr>
<td>MOV</td>
<td>ADDR OF COMMAND STRING</td>
</tr>
<tr>
<td>SUB</td>
<td>FIRST CHAR</td>
</tr>
<tr>
<td>CMP</td>
<td>0 TO 7</td>
</tr>
<tr>
<td>CLC</td>
<td>TOO SMALL</td>
</tr>
<tr>
<td>BJR</td>
<td>TOO BIG</td>
</tr>
<tr>
<td>BJR</td>
<td>ADDRESS OF DRIVE</td>
</tr>
<tr>
<td>BNR</td>
<td>NEXT INPUT CHAR</td>
</tr>
<tr>
<td>BNE</td>
<td>ALL DONE IF C.R.</td>
</tr>
<tr>
<td>BR</td>
<td>IS IT F?</td>
</tr>
<tr>
<td>BSR</td>
<td>NO SO ERROR</td>
</tr>
<tr>
<td>BLS</td>
<td>NEXT CHAR MUST BE C.R.</td>
</tr>
<tr>
<td>BSR</td>
<td>ELSE, ERROR</td>
</tr>
<tr>
<td>BNE</td>
<td>USE EVEN DRIVE IF RK-DFS</td>
</tr>
<tr>
<td>BNE</td>
<td>SHOW F TYPE DRIVE</td>
</tr>
</tbody>
</table>

Seg 0063
SOFTWARE SWITCH REGISTER IN USE?
BR IF NOT
REQUEST NEW CONTENTS FOR SWITCH REG
CONTINUE
WAIT FOR OPERATOR TO ENTER NEW SWR VALUE
HOLD SWITCHES
WAIT SW1 ONLY?
SW1 SET? SO LOW CYLINDER
F DRIVE?
NO
ODD DRIVE IF HIGH TRACK OF F
ADDRESS DRIVE
WRITE PROTECT
WAIT FOR DRIVE READY
RESET CONTROLLER
WAIT FOR READY
F DRIVE?
NO
TRACK 5 OF HIGH
SW1 SET?
YES SO TEST TRACK B OF DRIVE HIGH
TRACK 13D. IF SW1 SET
CYLINDER 64 IF NOT F
SW1 SET?
YES, SO CYLINDER 64
CYLINDER 105
ANY ERROR?
NO CONTINUE
RESET
WAIT FOR READY
SWITCH REG TO R2
SW1 ONLY
ANY CHANGE SINCE LAST?
YES, GO SET-UP ADDR AGAIN
ADDRESS THE DRIVE
WRITE PROTECT THE DRIVE
WAIT FOR CONTROL RDY
CLEAR TRACK ADDR
SWD SET?
NO TEST TRACK 0
TEST TRACK 1
CLEAR CYLINDER ADDR
PUT CYLINDER ADDR IN ADDR
ADDRESS THE DRIVE
READ I PROTON
INTO THIS BUFFER
READ, GO
DONE?
68TTL DISK POWER FAILURE TEST

POWER FAILURE (DURING DISK WRITE) TEST

The information written on the disk is lost when a loss of power occurs during a disk write operation. Upon entry the program finds out the disk ID (drives 0-9) and proceeds to test. Since the drives are written with unique patterns, then the heads must move to the cylinder base (decimal) and a message (DROP DISK) is displayed. After receiving this message, the user should remove power from the drive. On sensing a loss of power, the program asks the user to put the disk back in the drive. The errors (DPL) and a write-check is performed to check if the sectors on the disk (cylinders 0-9 and 10-15) are still there. If not a write-check error is reported.

; INITIALIZE DRIVE

; CONTROL RESET

; INITIALIZE PATTERN TO BE WRITTEN
; DISK POWER FAILURE TEST

; FILL THE PATTERN IN DATA BUFFER
; BUS ADDRES
; WRITE 1 CYL (256x128) WORDS:
; WRITE, GO, IBA SET
; WAIT FOR CONTROL READY

; DONE
; WRITING ALL 15 CYLINDERS:
; IF MOT, GO BAK

; DRIVE II:
; CYL 10
; PATTERN TO BE WRITTEN
; ADRES THE DISK
; WORD COUNT 1 CYLINDER
; BUS ADDRES
; WRITE, GO, IBA
; WAIT FOR THE HEADS TO SETTLE
; ON CYL 10
; TYPE ASCIIZ STRING
; GET OVER THE ASCIIZ

; DROP POWER

; ADRES THE DISK
; WRITING 14 CYLINDERS:
; BUS ADDRES
; WRITE, GO, IBA
; WAIT FOR CONTROL READY
; WAIT FOR DRIVE POWER TO GO DOWN
; OTHERWISE, KEEP ON WRITING ON CYL 10
; IF DRIVE POWER LOSS WAS SENSED,
; ASK TO PUT POWER ON
; TYPE ASCIIZ STRING
; GET OVER THE ASCIIZ

; WAIT FOR DRIVE READY
; CONTROL RESET, CLEAR ERROR

; INITIALIZE PATTERN
; WRITE CHECK, GO, IBA
;Horizonal Tab Processor

;This routine is used to change a 16-bit binary number to a 6-digit octal (ASCII) and type

;******************************************************************************
; This routine is used to change a 16-bit binary number to a 6-digit octal (ASCII) and type it.
;******************************************************************************

*STYPON---Enter here to type out with the same parameters as the last
*STYPOS OR STYPOC
*CALL:
  * MOV NUM, -(SP) ; NUMBER TO BE TYPED
  * TYPOS CALL FOR TYPEOUT
  * .BYTE N ; N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
  * .BYTE M ; M=1 OR 0
  * INC1 ; =TYPE LEADING ZEROS
  * D=SUPPRESS LEADING ZEROS

*STYPON---Enter here to type out with the same parameters as the last
*STYPOS OR STYPOC
*CALL:
  * MOV NUM, -(SP) ; NUMBER TO BE TYPED
**TYPOC----ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER**

**CALL:**

- MOV NUM,-(SP) ; NUMBER TO BE_TYPED
- TYPOC

**$TYPOS:**

- MOV 0(256),0(256) ; PICKUP THE MODE
- MOV 1(256),0(256) ; LOAD ZERO FILL SWITCH
- MOV 2(256),0(256) ; NUMBER OF DIGITS TO TYPE
- MOV 3(256),0(256) ; ADJUST RETURN ADDRESS
- MOV 4(256),0(256) ; SET THE ZERO FILL SWITCH
- MOV 5(256),0(256) ; SET FOR SIX(6) DIGITS
- MOV 6(256),0(256) ; SET THE ITERATION COUNT
- MOV 7(256),0(256) ; SAVE R3
- MOV 8(256),0(256) ; SUBTRACT IT FOR MAX. ALLOWED
- MOV 9(256),0(256) ; SAVE IT FOR USE
- MOV 10(256),0(256) ; GET THE ZERO FILL SWITCH
- MOV 11(256),0(256) ; PICKUP THE INPUT NUMBER
- MOV 12(256),0(256) ; CLEAR THE OUTPUT WORD
- MOV 13(256),0(256) ; FORM THIS DIGIT
- MOV 14(256),0(256) ; GET LSB OF THIS DIGIT
- MOV 15(256),0(256) ; TYPE THIS DIGIT?
- MOV 16(256),0(256) ; BR IF NO
- MOV 17(256),0(256) ; GET RID OF JUNK
- MOV 18(256),0(256) ; TEST FOR 0
- MOV 19(256),0(256) ; SUPPRESS THIS 0?
- MOV 20(256),0(256) ; BR IF YES
- MOV 21(256),0(256) ; DON'T SUPPRESS ANYMORE 0'S
- MOV 22(256),0(256) ; MAKE THIS DIGIT ASCII
- MOV 23(256),0(256) ; MAKE ASCII IF NOT ALREADY
- MOV 24(256),0(256) ; SAVE FOR TYING
- MOV 25(256),0(256) ; GO TYPE THIS DIGIT
- MOV 26(256),0(256) ; COUNT BY 1
- MOV 27(256),0(256) ; BR IF MORE TO DO
- MOV 28(256),0(256) ; BR IF DONE
- MOV 29(256),0(256) ; INSURE LAST DIGIT ISN'T A BLANK
- MOV 30(256),0(256) ; DO THE LAST DIGIT
- MOV 31(256),0(256) ; RESTORE R5
- MOV 32(256),0(256) ; RESTORE R4
- MOV 33(256),0(256) ; RESTORE R3
- MOV 34(256),0(256) ; SET THE STACK FOR RETURNING
- MOV 35(256),0(256) ; RETURN

**$BYTE 0**

- MOV 0(256),0(256) ; STORAGE FOR ASCII DIGIT
- MOV 1(256),0(256) ; TERMINATOR FOR TYPE ROUTINE
$OFILL:  BYTE  0  ; ZERO FILL SWITCH
$S1000:  WORD  0  ; NUMBER OF DIGITS TO TYPE

; TTY INPUT ROUTINE

; NUMBER OF ITEMS IN QUEUE
; INPUT POINTER
; OUTPUT POINTER
; TTY KEYBOARD QUEUE

; TK INITIALIZATION ROUTINE
; THIS ROUTINE WILL INITIALIZE THE TTY KEYBOARD INPUT QUEUE
; SETUP THE INTERRUPT VECTOR AND TURN ON THE KEYBOARD INTERRUPT

; CALL
; JSR  PC,$TINT
; RETURN

; TINT:  CLR  $TINT  ; CLEAR COUNT OF ITEMS IN QUEUE
; MOV  $TINT,$TQINT  ; MOVE THE STARTING ADDRESS OF THE
; MOV  $TQINT,$TQOUT  ; QUEUE INTO THE INPUT & OUTPUT POINTERS
; MOV  $TQINT,$TQVEC  ; INITIALIZE THE KEYBOARD VECTOR
; MOV  $TQINT,$TQVEC+2  ; BRK LEVEL 4
; TST  $QKB  ; CLEAR DONE FLAG
; RTS  $1000,$QTS  ; ENABLE TTY KEYBOARD INTERRUPT
; RTS  PC  ; RETURN 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.

; TKSERV:  MOVB  @TQKB,-(SP)  ; PICKUP THE CHARACTER
; CMP  (SP),#7  ; STRIP THE TTY
; BNE  2$  ; IS IT A CONTROL C?
; BGE  6$  ; BRANCH IF NO
; CMP  #SMREGB,SMR  ; IS SOFT-MMR SELECTED?
; BEQ  6$  ; GO TO SMR CHANGE

; 2$:  CMP  #TINT  ; IS THE QUEUE FULL?
; BNE  3$  ; BRANCH IF NO
; CMP  (SP),#23  ; RING THE TTY BELL
; BNE  5$  ; CLEAN CHARACTER OFF OF STACK
; TST  (SP)+  ; EXIT

; 3$:  CMP  (SP),#23  ; IS IT A CONTROL-5?
; BNE  5$  ; BRANCH IF NO
; CLR  $TINT  ; DISABLE TTY KEYBOARD INTERRUPTS
; TST  (SP)+  ; CLEAN CHAR OFF STACK
; BEQ  31$  ; WAIT FOR A CHAR
; CMP  #31  ; LOOP UNTIL ITS THERE
; MOVB  @TQKB,-(SP)  ; SET THE CHARACTER
IS IT A CONTROL-Q?
BRANCH IF NO

IS IT A CONTROL-G?
BRANCH IF YES

GET THE CONTROL-G AND SAVAGE THE TTY KEYBOARD INTERRUPTS
RETURN

IS IT A CONTROL-0?
BRANCH IF YES

IS IT A CONTROL-E?
BRANCH IF YES

MAKE IT UPPERCASE
UPDATE THE POINTER
GO OFF THE END
BRANCH IF NO
RESET THE POINTER
RETURN

SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP
CALL WHEN OPERATING IN TTY INTERRUPT MODE

IS THE SOFT-SWR SELECTED
EXIT IF NOT

IS A CHAR WAITING?
IF NOT, EXIT

MAKE IT 7-BIT ASCII
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-Q BEING TYPED, AND THE SOFTWARE SWITCH REGISTER BEING SELECTED.

ARE WE RUNNING IN AUTO-MODE?
BRANCH IF YES

CLEAR CONTROL-G OFF STACK
DISABLE TTY KEYBOARD INTERRUPTS
SET INTERRUPT MODE INDICATOR

ECHO THE CONTROL-G (1G)
TYPE CURRENT CONTENTS
THE NEW SR

PICK UP CHAR

H06

2666 023740 021507 000025 9$ CMP (SP),#25
2667 023724 001006 BNE 10%
2668 023746 104001 024072 TYPE $CNTRU
2669 023792 062706 000006 20% ADO #SP
2670 023356 000797 BR 19%

2671 023360 021507 000015 10% CMP (SP),#15
2672 023724 001006 BNE 168
2673 023746 012773 000004 TST (4),#0
2674 023792 018777 008356 BEQ 11%
2675 023737 018116 155536 MOV 2(SP),$ASWR
2676 023746 011316 000001 ADD #SP,0
2677 023724 001006 14%YPE $SLF
2678 023746 001006 14% CMPB #INTAG,#1
2679 023724 001006 BNE 156
2680 023746 001006 MOV #100, #STKS
2681 023792 001006 15% RFI PC $TYPEC
2682 023746 022440 000000 JSR (SP),#60
2683 023792 024040 000007 CMP (SP),#67
2684 023746 024040 000006 BLT 18%
2685 023724 001006 15% CMP (SP),#67
2686 023746 001006 15% CMP (SP),#67
2687 023724 001006 15% CMP (SP),#67
2688 023746 001006 15% CMP (SP),#67
2689 023724 001006 15% CMP (SP),#67
2690 023746 001006 15% CMP (SP),#67
2691 023724 001006 15% CMP (SP),#67
2692 023746 001006 15% CMP (SP),#67
2693 023724 001006 15% CMP (SP),#67
2694 023746 001006 15% CMP (SP),#67
2695 023724 001006 15% CMP (SP),#67
2696 023746 001006 15% CMP (SP),#67
2697 023724 001006 15% CMP (SP),#67
2698 023746 001006 15% CMP (SP),#67
2699 023724 001006 15% CMP (SP),#67
2700 023746 001006 15% CMP (SP),#67
2701 023724 001006 15% CMP (SP),#67
2702 023746 001006 15% CMP (SP),#67
2703 023724 001006 15% CMP (SP),#67
2704 023746 001006 15% CMP (SP),#67
2705 023724 001006 15% CMP (SP),#67
2706 023746 001006 15% CMP (SP),#67
2707 023724 001006 15% CMP (SP),#67
2708 023746 001006 15% CMP (SP),#67
2709 023724 001006 15% CMP (SP),#67
2710 023746 001006 15% CMP (SP),#67
2711 023724 001006 15% CMP (SP),#67
2712 023746 001006 15% CMP (SP),#67
2713 023724 001006 15% CMP (SP),#67
2714 023746 001006 15% CMP (SP),#67
2715 023724 001006 15% CMP (SP),#67
2716 023746 001006 15% CMP (SP),#67
2717 023724 001006 15% CMP (SP),#67
2718 023746 001006 15% CMP (SP),#67
2719 023724 001006 15% CMP (SP),#67
2720 023746 001006 15% CMP (SP),#67

$RDCHR: MOV (SP), - (SP)
2721 023512 011646 000004 000002 MOV (4), (SP)
2722 023517 011646 000004 MOV (4), (2(SP))
2723 023522 005046 CLR (SP)
2724 023530 001274 023536 MOV $64, (SP)
2725 023534 000002 RTI
2726 023536 005737 022736 64% TST $STKNT
2727 023536 001004 024072 TYPE $CNTRU
2728 023536 001006 BNE 168
2729 023536 001006 TST (4),#0
2730 023536 001006 BEQ 11%
2731 023536 001006 MOV 2(SP),$ASWR
2732 023536 001006 ADD #SP,0
2733 023536 001006 CMPB #INTAG,#1
2734 023536 001006 BNE 156
2735 023536 001006 MOV #100, #STKS

#THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY

CALL: RDCHR
RETURN HERE: PUSH DOWN THE PC AND
PROGRAM: IF THE PS IS READY FOR A CHARACTER
PUT NEW PS ON STACK
PUT NEW PC ON STACK
POP NEW PC AND PS

#SIMULATE CONTROL-U

#SIMULATE CONTROL-U

#SIMULATE CONTROL-U

#SIMULATE CONTROL-U
JOB

TTY INPUT ROUTINE

TYPE SLF
TST (SP)+
MOV (SP)+, R3
MOV (SP)+, (SP)
MOV 4(SP)+, (SP)
MOV #TTYIN, 4(SP)
RRI
9$:
BYTE 0
$TTYIN:
BYTE 30
$G2ELL:
ASCII (207)<(377)<(377)
$CNTL:
ASCII /U/<(15)<(12)
$MSWR:
ASCII (15)<(12)/SMR = /
$MNIEW:
ASCII / NEW = /

.SBTL READ AN OCTAL NUMBER FROM THE TTY

:*******************************
:THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
:CHANGE IT TO BINARY.
:CALL:
:* RDOCT
:* RETURN HERE
:* READ AN OCTAL NUMBER
:* HIGH ORDER BITS ARE IN S$IOCT

$RDOCT:
MOV (SP) -(SP)
MOV 4(SP), (SP)
MOV R0, -(SP)
MOV R1, -(SP)
MOV R2, -(SP)
1$:
ROL (SP)+, R0
MOV R1, R1
CLR R2
CLR R2
2$:
MOVB (RO)+, -(SP)
BEQ 3$
ASL R1
ROL R2
ASL R1
ROL R2
ASL R1
ROL R2
3$:
TST (SP)+
MOV R1, R1(SP)
MOV R2, S$IOCT
MOV (SP)+, R2
MOV (SP)+, R1
MOV (SP)+, R0
RRI
$SIOCT
.WORD 0
K06

:;THIS ROUTINE IS TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
:;OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
:;GO TO THAT ROUTINE.

$TRAP: MOV RO ,-(SP) ;SAVE RO
     MOV 2(SP),RO ;GET TRAP ADDRESS
     IST (RO) ;BACKUP BY 2
     MOV B (RO),RO ;GET RIGHT BYTE OF TRAP
     ASL RO ;POSITION FOR INDEXING
     MOV $TRPAD(RO),RO ;INDEX TO TABLE
     RTS RO ;GO TO ROUTINE

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

$TRAP2: MOV (SP),-(SP) ;MOV THE PC DOWN
      MOV 4(SP),2(SP) ;MOVE THE PSW DOWN
      RTI ;RESTORE THE PSW

.SBTTL TRAP TABLE

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

:ROUTINE

:;TRPAD:

  $rtc:
  $type: ;CALL=TYPE
  $trap+: TRAP+(104001) TTY TYPEOUT ROUTINE
  $type: ;CALL=TYPE
  $trap+: TRAP+(104002) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
  $type: ;CALL=TYPE
  $trap+: TRAP+(104003) TYPE OCTAL NUMBER (NO LEADING ZEROS)
  $type: ;CALL=TYPE
  $trap+: TRAP+(104004) TYPE OCTAL NUMBER (AS PER LAST CALL)

  $gtsw: ;CALL=GTSW
  $trap+: TRAP+(104005) GET SOFT-SWR SETTING

  $cksw: ;CALL=CKSW
  $trap+: TRAP+(104006) TEST FOR CHANGE IN SOFT-SWR

  $rchr: ;CALL=RCHR
  $trap+: TRAP+(104007) TTY TYPE IN CHARACTER ROUTINE

  $rdlin: ;CALL=RDLIN
  $trap+: TRAP+(104008) TTY TYPE IN STRING ROUTINE

  $rdoct: ;CALL=RODOCT
  $trap+: TRAP+(11104011) READ AN OCTAL NUMBER FROM TTY

.SBTTL POWER DOWN AND UP ROUTINES

:POWER DOWN ROUTINE

$power: MOV $illup,28h

:POWER UP ROUTINE

$power: MOV $illup,0

:;SAVE SP
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALTS</td>
<td>002644</td>
</tr>
<tr>
<td>BENS</td>
<td>002713</td>
</tr>
<tr>
<td>BONIN</td>
<td>002713</td>
</tr>
<tr>
<td>BONOE</td>
<td>012403</td>
</tr>
<tr>
<td>BONON</td>
<td>012403</td>
</tr>
<tr>
<td>BONMC</td>
<td>012403</td>
</tr>
<tr>
<td>BONCE</td>
<td>012403</td>
</tr>
<tr>
<td>BONCT</td>
<td>012403</td>
</tr>
<tr>
<td>BIT0</td>
<td>000100</td>
</tr>
<tr>
<td>BIT1</td>
<td>000100</td>
</tr>
<tr>
<td>BIT2</td>
<td>000004</td>
</tr>
<tr>
<td>BIT3</td>
<td>000004</td>
</tr>
<tr>
<td>BIT4</td>
<td>000004</td>
</tr>
<tr>
<td>BIT5</td>
<td>000004</td>
</tr>
<tr>
<td>BIT6</td>
<td>000004</td>
</tr>
<tr>
<td>BIT7</td>
<td>000004</td>
</tr>
<tr>
<td>BIT8</td>
<td>000004</td>
</tr>
<tr>
<td>BIT9</td>
<td>000004</td>
</tr>
<tr>
<td>BIT10</td>
<td>000004</td>
</tr>
<tr>
<td>BIT11</td>
<td>000004</td>
</tr>
<tr>
<td>BIT12</td>
<td>000004</td>
</tr>
<tr>
<td>BIT13</td>
<td>000004</td>
</tr>
<tr>
<td>BIT14</td>
<td>000004</td>
</tr>
<tr>
<td>BIT15</td>
<td>000004</td>
</tr>
<tr>
<td>BIT16</td>
<td>000004</td>
</tr>
<tr>
<td>BIT17</td>
<td>000004</td>
</tr>
<tr>
<td>BIT18</td>
<td>000004</td>
</tr>
<tr>
<td>BIT19</td>
<td>000004</td>
</tr>
<tr>
<td>BUFR</td>
<td>017000</td>
</tr>
<tr>
<td>BUSDR</td>
<td>001336</td>
</tr>
<tr>
<td>CHKIN</td>
<td>006450</td>
</tr>
<tr>
<td>CHKIN</td>
<td>006450</td>
</tr>
<tr>
<td>CKSFR</td>
<td>004006</td>
</tr>
<tr>
<td>CRDPL</td>
<td>020083</td>
</tr>
<tr>
<td>CNTSIN</td>
<td>001322</td>
</tr>
<tr>
<td>COMON</td>
<td>01230</td>
</tr>
<tr>
<td>CONM</td>
<td>012314</td>
</tr>
<tr>
<td>CONTN</td>
<td>011409</td>
</tr>
<tr>
<td>CTNL</td>
<td>001322</td>
</tr>
<tr>
<td>CR</td>
<td>000015</td>
</tr>
<tr>
<td>CRLF</td>
<td>000200</td>
</tr>
<tr>
<td>CYLRC</td>
<td>004320</td>
</tr>
</tbody>
</table>

User Symbols Cross Reference Table
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXIT</td>
<td>64</td>
</tr>
<tr>
<td>XTR</td>
<td>550</td>
</tr>
<tr>
<td>SC</td>
<td>549</td>
</tr>
<tr>
<td>FIL</td>
<td>549</td>
</tr>
<tr>
<td>XTR2</td>
<td>1397</td>
</tr>
<tr>
<td>XTRX</td>
<td>1444</td>
</tr>
<tr>
<td>XTRX2</td>
<td>1490</td>
</tr>
<tr>
<td>2226</td>
<td>2236</td>
</tr>
<tr>
<td>2278</td>
<td></td>
</tr>
<tr>
<td>SC1</td>
<td>000612</td>
</tr>
<tr>
<td>G007</td>
<td>001241</td>
</tr>
<tr>
<td>G003</td>
<td>001241</td>
</tr>
<tr>
<td>G013</td>
<td>001241</td>
</tr>
<tr>
<td>G048</td>
<td>001241</td>
</tr>
<tr>
<td>INDEX</td>
<td>001241</td>
</tr>
<tr>
<td>INSTR</td>
<td>001241</td>
</tr>
<tr>
<td>D007</td>
<td>001241</td>
</tr>
<tr>
<td>IOVEC</td>
<td>001338</td>
</tr>
<tr>
<td>KYTEF</td>
<td>001338</td>
</tr>
<tr>
<td>DFLC</td>
<td>004212</td>
</tr>
<tr>
<td>DSEEK</td>
<td>004212</td>
</tr>
<tr>
<td>LF</td>
<td>000012</td>
</tr>
<tr>
<td>LOGA</td>
<td>001144</td>
</tr>
<tr>
<td>MANSEL</td>
<td>010615</td>
</tr>
<tr>
<td>MASKER</td>
<td>012312</td>
</tr>
<tr>
<td>MOUNT</td>
<td>005316</td>
</tr>
<tr>
<td>MSK1DL</td>
<td>005316</td>
</tr>
<tr>
<td>NR1</td>
<td>002127</td>
</tr>
<tr>
<td>NR2</td>
<td>017662</td>
</tr>
<tr>
<td>NR3</td>
<td>1890</td>
</tr>
<tr>
<td>NR4</td>
<td>1905</td>
</tr>
<tr>
<td>NR5</td>
<td>1908</td>
</tr>
<tr>
<td>NR6</td>
<td>1921</td>
</tr>
<tr>
<td>Symbol</td>
<td>Value</td>
</tr>
<tr>
<td>--------</td>
<td>-------</td>
</tr>
<tr>
<td>ABC</td>
<td>374</td>
</tr>
<tr>
<td>DEF</td>
<td>278</td>
</tr>
<tr>
<td>GHI</td>
<td>498</td>
</tr>
<tr>
<td>JKL</td>
<td>519</td>
</tr>
<tr>
<td>MNO</td>
<td>498</td>
</tr>
<tr>
<td>PQR</td>
<td>518</td>
</tr>
</tbody>
</table>

**Note:** The table continues with similar entries that are not fully visible in the image.
<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>XA</td>
<td>2875</td>
</tr>
<tr>
<td>ZL</td>
<td>2736</td>
</tr>
<tr>
<td>XDO</td>
<td>2873</td>
</tr>
<tr>
<td>XDD</td>
<td>2805</td>
</tr>
<tr>
<td>XDDP</td>
<td>2874</td>
</tr>
<tr>
<td>XDP</td>
<td>2804</td>
</tr>
<tr>
<td>XDS</td>
<td>2845</td>
</tr>
<tr>
<td>XDSQ</td>
<td>2914</td>
</tr>
<tr>
<td>XDSU</td>
<td>2954</td>
</tr>
<tr>
<td>XDSV</td>
<td>2955</td>
</tr>
<tr>
<td>XDSW</td>
<td>2591</td>
</tr>
<tr>
<td>XDSX</td>
<td>2596</td>
</tr>
<tr>
<td>XDSY</td>
<td>2597</td>
</tr>
<tr>
<td>XDSZ</td>
<td>2627</td>
</tr>
<tr>
<td>XDAA</td>
<td>2795</td>
</tr>
<tr>
<td>XDB</td>
<td>2896</td>
</tr>
<tr>
<td>XDBA</td>
<td>2897</td>
</tr>
<tr>
<td>XDBB</td>
<td>2998</td>
</tr>
<tr>
<td>XDBC</td>
<td>2914</td>
</tr>
<tr>
<td>XDBC</td>
<td>354</td>
</tr>
<tr>
<td>XDBD</td>
<td>355</td>
</tr>
<tr>
<td>XDBE</td>
<td>2591</td>
</tr>
<tr>
<td>XDBF</td>
<td>2596</td>
</tr>
<tr>
<td>XDBG</td>
<td>2597</td>
</tr>
<tr>
<td>XDBH</td>
<td>2627</td>
</tr>
<tr>
<td>XDBI</td>
<td>2795</td>
</tr>
<tr>
<td>XDBJ</td>
<td>331</td>
</tr>
<tr>
<td>XDBK</td>
<td>333</td>
</tr>
<tr>
<td>XDBL</td>
<td>354</td>
</tr>
<tr>
<td>XDBM</td>
<td>355</td>
</tr>
<tr>
<td>XDBN</td>
<td>2591</td>
</tr>
<tr>
<td>XDBO</td>
<td>2596</td>
</tr>
<tr>
<td>XDBP</td>
<td>2597</td>
</tr>
<tr>
<td>XDBQ</td>
<td>2627</td>
</tr>
<tr>
<td>XDBR</td>
<td>2795</td>
</tr>
<tr>
<td>XDSS</td>
<td>331</td>
</tr>
<tr>
<td>XDST</td>
<td>333</td>
</tr>
<tr>
<td>XDSU</td>
<td>354</td>
</tr>
<tr>
<td>XDSV</td>
<td>355</td>
</tr>
<tr>
<td>XDSW</td>
<td>2591</td>
</tr>
<tr>
<td>XDSX</td>
<td>2596</td>
</tr>
<tr>
<td>XDSY</td>
<td>2597</td>
</tr>
<tr>
<td>XDSZ</td>
<td>2627</td>
</tr>
<tr>
<td>XDA</td>
<td>2795</td>
</tr>
<tr>
<td>XDAA</td>
<td>2896</td>
</tr>
<tr>
<td>XDAA</td>
<td>2897</td>
</tr>
<tr>
<td>XDAF</td>
<td>2914</td>
</tr>
<tr>
<td>XDAG</td>
<td>354</td>
</tr>
<tr>
<td>XDAH</td>
<td>355</td>
</tr>
<tr>
<td>XDAI</td>
<td>2591</td>
</tr>
<tr>
<td>XDAJ</td>
<td>2596</td>
</tr>
<tr>
<td>XDAK</td>
<td>2597</td>
</tr>
<tr>
<td>XDAL</td>
<td>2627</td>
</tr>
<tr>
<td>XDAM</td>
<td>2795</td>
</tr>
<tr>
<td>XDAN</td>
<td>331</td>
</tr>
<tr>
<td>XDAP</td>
<td>333</td>
</tr>
<tr>
<td>XDAR</td>
<td>354</td>
</tr>
<tr>
<td>XDAS</td>
<td>355</td>
</tr>
<tr>
<td>XDAT</td>
<td>2591</td>
</tr>
<tr>
<td>XDBT</td>
<td>2596</td>
</tr>
<tr>
<td>XDBU</td>
<td>2597</td>
</tr>
<tr>
<td>XDBV</td>
<td>2627</td>
</tr>
<tr>
<td>XDBW</td>
<td>2795</td>
</tr>
<tr>
<td>XDBX</td>
<td>331</td>
</tr>
<tr>
<td>XDBY</td>
<td>333</td>
</tr>
<tr>
<td>XDBZ</td>
<td>354</td>
</tr>
<tr>
<td>XDC</td>
<td>355</td>
</tr>
<tr>
<td>XDCS</td>
<td>2591</td>
</tr>
<tr>
<td>XDCT</td>
<td>2596</td>
</tr>
<tr>
<td>XDCU</td>
<td>2597</td>
</tr>
<tr>
<td>XDCV</td>
<td>2627</td>
</tr>
<tr>
<td>XDCW</td>
<td>2795</td>
</tr>
<tr>
<td>XDDE</td>
<td>331</td>
</tr>
<tr>
<td>XDDF</td>
<td>333</td>
</tr>
<tr>
<td>XDNG</td>
<td>354</td>
</tr>
<tr>
<td>XDNH</td>
<td>355</td>
</tr>
<tr>
<td>XDNI</td>
<td>2591</td>
</tr>
<tr>
<td>XDNJ</td>
<td>2596</td>
</tr>
<tr>
<td>XDNK</td>
<td>2597</td>
</tr>
<tr>
<td>XDNL</td>
<td>2627</td>
</tr>
<tr>
<td>XDNS</td>
<td>2795</td>
</tr>
<tr>
<td>XDNO</td>
<td>331</td>
</tr>
<tr>
<td>XDNF</td>
<td>333</td>
</tr>
<tr>
<td>XDNG</td>
<td>354</td>
</tr>
<tr>
<td>XDNH</td>
<td>355</td>
</tr>
<tr>
<td>XDNJ</td>
<td>2591</td>
</tr>
<tr>
<td>XDNK</td>
<td>2596</td>
</tr>
<tr>
<td>XDNL</td>
<td>2597</td>
</tr>
<tr>
<td>XDNS</td>
<td>2627</td>
</tr>
<tr>
<td>XDNF</td>
<td>2795</td>
</tr>
<tr>
<td>XDN</td>
<td>331</td>
</tr>
<tr>
<td>XDNN</td>
<td>333</td>
</tr>
<tr>
<td>XDNG</td>
<td>354</td>
</tr>
<tr>
<td>XDNH</td>
<td>355</td>
</tr>
<tr>
<td>XDNJ</td>
<td>2591</td>
</tr>
<tr>
<td>XDNK</td>
<td>2596</td>
</tr>
<tr>
<td>XDNL</td>
<td>2597</td>
</tr>
<tr>
<td>XDNS</td>
<td>2627</td>
</tr>
<tr>
<td>XDNF</td>
<td>2795</td>
</tr>
<tr>
<td>XDN</td>
<td>331</td>
</tr>
<tr>
<td>XDNN</td>
<td>333</td>
</tr>
<tr>
<td>XDNG</td>
<td>354</td>
</tr>
<tr>
<td>XDNH</td>
<td>355</td>
</tr>
<tr>
<td>XDNJ</td>
<td>2591</td>
</tr>
<tr>
<td>XDNK</td>
<td>2596</td>
</tr>
<tr>
<td>XDNL</td>
<td>2597</td>
</tr>
<tr>
<td>XDNS</td>
<td>2627</td>
</tr>
<tr>
<td>XDNF</td>
<td>2795</td>
</tr>
<tr>
<td>XDN</td>
<td>331</td>
</tr>
<tr>
<td>XDNN</td>
<td>333</td>
</tr>
<tr>
<td>XDNG</td>
<td>354</td>
</tr>
<tr>
<td>XDNH</td>
<td>355</td>
</tr>
<tr>
<td>XDNJ</td>
<td>2591</td>
</tr>
<tr>
<td>XDNK</td>
<td>2596</td>
</tr>
<tr>
<td>XDNL</td>
<td>2597</td>
</tr>
<tr>
<td>XDNS</td>
<td>2627</td>
</tr>
<tr>
<td>XDNF</td>
<td>2795</td>
</tr>
<tr>
<td>XDN</td>
<td>331</td>
</tr>
<tr>
<td>XDNN</td>
<td>333</td>
</tr>
<tr>
<td>XDNG</td>
<td>354</td>
</tr>
<tr>
<td>XDNH</td>
<td>355</td>
</tr>
<tr>
<td>XDNJ</td>
<td>2591</td>
</tr>
<tr>
<td>XDNK</td>
<td>2596</td>
</tr>
<tr>
<td>XDNL</td>
<td>2597</td>
</tr>
<tr>
<td>XDNS</td>
<td>2627</td>
</tr>
<tr>
<td>XDNF</td>
<td>2795</td>
</tr>
</tbody>
</table>
CROSS REFERENCE TABLE -- MACRO NAMES

<table>
<thead>
<tr>
<th>MACRO</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDIV</td>
<td>1</td>
</tr>
<tr>
<td>SEOP</td>
<td>1</td>
</tr>
<tr>
<td>SERR</td>
<td>1</td>
</tr>
<tr>
<td>SERRT</td>
<td>1</td>
</tr>
<tr>
<td>SPSMT</td>
<td>1</td>
</tr>
<tr>
<td>SPOE</td>
<td>2875</td>
</tr>
<tr>
<td>SREDO</td>
<td>2795</td>
</tr>
<tr>
<td>SPDRE</td>
<td>2556</td>
</tr>
<tr>
<td>SRED</td>
<td>2556</td>
</tr>
<tr>
<td>SREP</td>
<td>2833</td>
</tr>
<tr>
<td>STYPB</td>
<td>2409</td>
</tr>
<tr>
<td>STYPO</td>
<td>2479</td>
</tr>
<tr>
<td>SYOCA</td>
<td>1</td>
</tr>
</tbody>
</table>

ABS. 024466 000

ERRORS DETECTED: 0

DZAKIF.BIN, DZAKIF.LST/CRF/SOL=DZAKKE.SML,DZAKIF.P11

RUN-TIME: 12.16 SECONDS
RUN-TIME RATIO: 102/31=3.3
CORE USED: 33K (65 PAGES)