

RP11E

FUNCT LOGIC + READ/WRITE
MD-11-DZRPY-C

EP-DZRPY-C-DL-C
COPYRIGHT © 75-77
FICHE 1 OF 1

DEC 1977
digital
MADE IN USA

The main body of the document is a large grid of 100 small diagrams, arranged in 10 rows and 10 columns. Each diagram appears to be a technical drawing, possibly a logic diagram or a data table, related to the 'FUNCT LOGIC + READ/WRITE' mentioned in the header. The diagrams are too small to read clearly, but they seem to be organized in a systematic way, possibly representing different components or states of the system. The diagrams are arranged in a grid that is 10 rows high and 10 columns wide. The diagrams are too small to read clearly, but they seem to be organized in a systematic way, possibly representing different components or states of the system.

CONTENTS

1. ABSTRACT
2. REQUIREMENTS
 - 2.1 EQUIPMENT
 - 2.2 STORAGE
 - 2.3 PRELIMINARY PROGRAMS
3. STARTING PROCEDURE
 - 3.1 LOADING PROCEDURES
 - 3.2 STARTING ADDRESSES
 - 3.3 OPERATOR ACTION
 - 3.4 'CONTROL C' OPTION
 - 3.5 DEFAULT RPIIE UNIBUS ADDRESS
4. OPERATING PROCEDURES
 - 4.1 OPERATIONAL SWITCH SETTINGS
 - 4.2 TELETYPE KEYBOARD AND PRINTER UNIBUS ADDRESSES
 - 4.3 TTY KEYBOARD ENTRY CONTROL CODES AND CONVENTIONS
 - 4.4 RPIIE UNIBUS ADDRESS CHANGE ROUTINE
 - 4.5 CONVERSATION MODE
 - 4.6 OPERATION OF TESTS 15, 16, & 17
5. ERROR HANDLING
 - 5.1 ERROR REPORTING
 - 5.2 ERROR TYPES
6. RESTRICTIONS
 - 6.1 SUBSYSTEMS WITH MIXED DRIVE TYPES
 - 6.2 FORMATTED PACKS
 - 6.3 'CONTROL C' TERMINATION DURING TESTS 5 & 13
 - 6.4 CONTROLLER INTERRUPT
7. MISCELLANEOUS
 - 7.1 RUN TIME
 - 7.2 STACK POINTER
 - 7.3 END OF PASS/END OF TEST
 - 7.4 SUBROUTINE CALLS
 - 7.5 DISK SURFACE USAGE
 - 7.6 TEST EXECUTION SEQUENCE
 - 7.7 DRIVE STATUS TIMEOUT
 - 7.8 DATA COMPARISON USING MEMORY MANAGEMENT
 - 7.9 PATTERN SELECTION WORD
 - 7.10 CONTROLLER/DRIVE OPERATION TIME LIMIT
8. TEST DESCRIPTIONS
9. PROGRAM LISTING

1. ABSTRACT

THE RPIIE FUNCTIONAL LOGIC & READ/WRITE TEST PERFORMS A CURSORY CHECK OF THE RPIIE CONTROLLED LOGIC AND PERFORMS DATA STORAGE/RETRIEVAL TESTING OF THE SUBSYSTEM. INCLUDED IN THE PROGRAM ARE UTILITY TESTS WHICH ASSIST IN PERFORMING HEAD ALIGNMENT, WHICH CHECK SUBSYSTEM POWER FAIL OPERATION, AND WHICH CHECK THE RPIIE AND DRIVE CONTROL SWITCHES.

THE PROGRAM CONTAINS A 'CONVERSATION MODE' WHICH ALLOWS THE OPERATOR TO MODIFY THE OPERATION PARAMETERS FOR THE DATA STORAGE/RETRIEVAL TEST. THE PROGRAM WILL USE MEMORY MANAGEMENT IN THE OPERATION OF THE DATA TEST IF THE PROGRAM IS BEING RUN ON A MEMORY MANAGEMENT SYSTEM.

THIS PROGRAM COMPLETES THE TESTING STARTED IN THE DISKLESS CONTROLLER TEST (MD-11-DZRPW). TESTING IN THE FUNCTIONAL LOGIC PROGRAM IS PERFORMED WITH 1 - 8 RPO2, RPRO2, OR RPO3 DISK DRIVES.

2. REQUIREMENTS

2.1 EQUIPMENT

PDP-11 COMPUTER WITH 12K OF MEMORY; CONSOLE TELETYPE; RPIIE DISK CONTROLLER; 1 TO 8 RPO2, RPRO2, OR RPO3 DISK DRIVES WITH FORMATTED PACKS.

2.2 STORAGE

THIS PROGRAM WILL LOAD AND RUN IN 12K.

2.3 PRELIMINARY PROGRAMS

MAINDEC-11-DZRPW - RPIIE DISKLESS LOGIC TEST

3. STARTING PROCEDURES

3.1 LOADING PROCEDURES

THE PROGRAM MAY BE LOADED FROM EITHER PAPER TAPE OR AN 'XXDP' DEVICE. REFER TO EITHER THE STANDARD PROCEDURES FOR LOADING 'ABS' FORMAT PAPER TAPES OR TO THE 'XXDP' SYSTEM REFERENCE DOCUMENT.

THIS PROGRAM MAY BE INCLUDED IN AN 'XXDP' CHAIN. IF THE PROGRAM IS PART OF A CHAIN AND IS LOADED FROM AN RPO2, RPRO2, OR AN RPO3, THE PROGRAM WILL NOT USE DRIVE C FOR TESTING.

3.2 STARTING ADDRESSES

200 NORMAL STARTING ADDRESS
 204 SELECT OPERATING PARAMETERS
 210 SELECT RP11 ADDRESS
 214 COMBINATION OF 204 AND 210

3.2.1 START FROM LOCATION 200

WHEN THE PROGRAM IS STARTED FROM LOCATION 200, ALL ONLINE DRIVES WILL BE USED (SEE SECTION 3.1 FOR DRIVE 0 EXCEPTION) USING TESTS 0 - 14. ALL PARAMETERS FOR TEST 14 WILL BE RETURNED TO THEIR DEFAULT VALUES.

3.2.2 START FROM LOCATION 204

WHEN THE PROGRAM IS STARTED FROM LOCATION 204, PARAMETERS FOR TEST 14 ARE RESET TO THEIR DEFAULT VALUES, AND THE PROGRAM ENTERS CONVERSATIONAL MODE.

3.2.3 START FROM LOCATION 210

WHEN THE PROGRAM IS STARTED FROM LOCATION 210, OPERATION IS THE SAME AS THE START FROM 200, EXCEPT THAT THE PROGRAM ASKS FOR AN RP11 ADDRESS. THE OPERATOR MAY ENTER A NEW ADDRESS VALUE OR HE MAY RETAIN THE CURRENT VALUE. THE PARAMETERS FOR TEST 14 ARE RESET TO THEIR DEFAULT VALUES.

3.2.4 START FROM LOCATION 214

PROGRAM START FROM LOCATION 214 IS THE SAME AS THE START FROM LOCATION 204 EXCEPT THAT THE PROGRAM ASKS FOR A NEW RP11 ADDRESS. THE PARAMETERS FOR TEST 14 ARE RESET TO THEIR DEFAULT VALUES.

3.3 OPERATOR ACTION

BEFOR STARTING REFER TO SECTION 4.1

1. LOAD PROGRAM INTO MEMORY (SEE SECTION 3.1)
2. CHANGE TELETYPE KEYBOARD AND PRINTER ADDRESSES AS REQUIRED (SEE SECTION 4.2)
3. PLACE A FORMATTED PACK ON DRIVE(S) TO BE USED
4. BRING DRIVE(S) TO ONLINE STATE
5. PLACE 'FORMAT' AND 'MAINTENANCE' SWITCHES ON THE RP11 CONTROL PANEL TO 'NORMAL'
6. SET CONTROLLER AND DRIVE WRITE LOCKOUT SWITCHES AS REQUIRED
7. LOAD ADDRESS 200, 204, 210, OR 214.
8. SET CPU SWITCH REGISTER SWITCHES (SEE SECTION 4.1)
9. PRESS START ON THE CPU CONTROL PANEL

3.4 'CONTROL C' OPTION

THE OPERATOR MAY TERMINATE THE CURRENT TEST AT ANY TIME BY TYPING A 'CONTROL C' ON THE TELETYPE KEYBOARD. TYPING A 'CONTROL C' CAUSES THE PROGRAM TO ENTER THE 'CONVERSATION MODE' ROUTINE. ALL THE PARAMETERS FOR TEST 14 WILL BE RETURNED TO THEIR DEFAULT VALUES.

3.5 DEFAULT RP11E UNIBUS ADDRESS

THE DEFAULT RP11E UNIBUS ADDRESS IS 176710. WHEN THE PROGRAM IS STARTED FROM EITHER LOCATION 200 OR LOCATION 204, ADDRESS 176710 WILL BE CHECKED AND IF NO RESPONSE IS RECEIVED FROM THAT ADDRESS THE PROGRAM WILL ENTER THE RP11E ADDRESS CHANGE ROUTINE. ADDRESS CHANGE OPERATION WILL BE IDENTICAL TO THE OPERATION WHEN THE PROGRAM IS STARTED FROM EITHER LOCATION 210 OR 214. (REFER TO SECTION 4.4)

- OPERATING PROCEDURES

IF THE DIAGNOSTIC IS RUN ON A CPU WITHOUT A SWITCH REGISTER THEN A SOFTWARE SWITCH REGISTER IS USED WHICH ALLOWS THE USER THE SAME SWITCH OPTIONS AS THE HARDWARE SWITCH REGISTER. IF THE HARDWARE SWITCH REGISTER DOES NOT EXIST OR IF ONE DOES AND IT CONTAINS ALL ONES (177777) THEN THE SOFTWARE SWITCH REGISTER (LOC. 176) IS USED.

CONTROL:

THIS PROGRAM ALSO SUPPORTS THE DYNAMIC LOADING OF THE SOFTWARE SWITCH REGISTER (LOC. 176) FROM THE TTY. THIS CAN BE ACCOMPLISHED BY DOING THE FOLLOWING:

- 1) TYPE CONTROL G (<G>); THIS WILL ALLOW THE TTY TO ENTER DATA INTO LOC. 176 AT SELECTED POINTS WITHIN THE PROGRAM.
- 2) THE MACHINE WILL THEN TYPE: SWR=XXXXXXNEW= (XXXXXX IS THE OCTAL CONTENTS OF THE SOFTWARE SWITCH REGISTER.)
3. AFTER THE 'NEW=' HAS BEEN TYPED THEN THE OPERATOR CAN DO ONE OF THE FOLLOWING AT THE TTY:
 - A) TYPE A NUMBER TO BE LOADED INTO LOC. 176 FOLLOWED BY A <CR>. (ONLY NUMBERS BETWEEN 0-7 WILL BE ACCEPTED AND ONLY 6 NUMBERS WILL BE ALLOWED)
IF A <CR> IS THE FIRST KEY DEPRESSED THE SOFTWARE SWITCH REGISTER CONTENTS WILL NOT BE CHANGED.
 - B) IF A CONTROL U (<U>) IS DEPRESSED THEN THE PROGRAM WILL SEND YOU BACK TO STEP 2.

4.1 OPERATIONAL SWITCH SETTINGS

WITH SW<15:0>=1. THE PROGRAM WILL TYPEOUT ERRORS AND CONTINUE WITH THE TEST.

THE SWITCH SETTINGS ARE:

- SW<15>=1...HALT ON ERROR
- SW<14>=1...LOOP ON TEST
- SW<13>=1...INHIBIT ERROR TYPEOUTS
- SW<11>=1...INHIBIT ITERATIONS
- SW<10>=1...RING BELL ON ERROR
- SW<09>=1...LOOP ON ERROR
- SW<07>=1...DON'T TYPE DRIVE STATUS WHEN PROGRAM STARTED OR WHEN ENTERING CONVERSATION MODE.
- SW<06>=1...DON'T USE MEMORY MANAGEMENT IN TEST 14
- SW<05>=1...CHANGE HEAD SELECTION IN TEST 16
- SW<04>=1...DISPLAY ALL READ RETRY ATTEMPTS IN TEST 14
- SW<03>=1...TYPE ALL DATA COMPARE ERRORS IN TESTS 13 & 14
- SW<02>=1...INHIBIT DATA COMPARE IN TESTS 13 & 14
- SW<01>=1...BYPASS READ & DATA COMPARE IN TEST 14
- SW<00>=1...BYPASS WRITE CHECK IN TEST 14

4.2 TELETYPE KEYBOARD AND PRINTER UNIBUS ADDRESSES

THE PROGRAM ASSUMES THE FOLLOWING UNIBUS ADDRESSES. THESE ADDRESSES MAY BE CHANGED AT THE INDICATED LOCATION BEFORE STARTING THE PROGRAM.

MEMORY LOCATION	CONTENTS	FUNCTION
1136	177560	TTY KEYBOARD STATUS REG
1140	177562	TTY KEYBOARD BUFFER REG
1142	177564	TTY PRINTER STATUS REGISTER
1144	177566	TTY PRINTER BUFFER REG

(THE RP11E UNIBUS ADDRESS IS CHANGED IN RESPONSE TO THE PROGRAM'S TYPEL REQUEST IF THE PROGRAM IS STARTED FROM EITHER LOCATION 210 OR 214 OR IF THERE IS NO RESPONSE WHEN LOCATION 176710 IS ADDRESSED.)

4.3 TTY KEYBOARD ENTRY CONTROL CODES AND CONVENTIONS

THE FOLLOWING CONTROL CODES AND CONVENTIONS ARE USED BY THE PROGRAM:

CARRIAGE RETURN (CR) - ALL ENTRIES MUST BE TERMINATED BY A CARRIAGE RETURN.

CONTROL U (^U) - IF THIS CONTROL CODE IS TYPED BEFORE AN ENTRY IS TERMINATED, THE ENTRY IS CLEARED AND ANOTHER VALUE MAY BE ENTERED. A ^U IS TYPED BY PRESSING THE 'CONTROL' AND 'U' KEYS TOGETHER.

RUBOUT (RO) - IF THIS CONTROL CODE IS TYPED, THE CHARACTER TYPED BEFORE THE 'RO' WILL BE DELETED FROM THE ENTRY. EACH TIME A 'RO' IS TYPED, THE LAST CHARACTER IN THE INPUT ENTRY IS DELETED. WHEN A 'RO' IS TYPED, THE ENTRY ROUTINE WILL TYPE A BACKSLASH ('\') CHARACTER AND THE CHARACTER BEING DELETED. WHEN A CHARACTER OTHER THAN A 'RO' IS TYPED, THE ENTRY ROUTINE WILL TYPE ANOTHER BACKSLASH CHARACTER AND TYPE THE NEW CHARACTER ENTERED. THE 'RO' CANNOT BE USED TO DELETE A 'CR'.

CONTROL C (IC) - THE OPERATION OF THE 'IC' CHARACTER IS DESCRIBED IN SECTION 3.4. 'IC' MAY ALSO BE USED TO ABORT THE CONVERSATION MODE AND TO RETURN TO THE BEGINNING OF CONVERSATION MODE. 'IC' IS GENERATED BY PRESSING THE 'CONTROL' KEY AND THE 'C' KEY.

IF ONLY A 'CR' IS ENTERED IN RESPONSE TO AN ENTRY REQUEST FROM THE PROGRAM, THE PROGRAM WILL USE THE DEFAULT VALUE FOR THE REQUESTED ENTRY.

LEADING ZEROS ARE NOT REQUIRED ON ANY ENTRY.

4.4 RP11E UNIBUS ADDRESS CHANGE ROUTINE

WHEN THE PROGRAM IS STARTED FROM LOCATION 200 OR LOCATION 204, THE PROGRAM WILL ASSUME THAT THE RP11E IS AT UNIBUS ADDRESS 176710. WHEN THE PROGRAM IS STARTED FROM EITHER LOCATION 213 OR LOCATION 214, THE PROGRAM ENTERS A ROUTINE WHICH ALLOWS THE OPERATOR TO CHANGE THE RP11E UNIBUS ADDRESS TO BE USED BY THE PROGRAM.

WHEN THE ADDRESS CHANGE ROUTINE IS ENTERED, THE PROGRAM TYPES THE PRESENT RP11E UNIBUS ADDRESS:

RPADR = AAAAAA/

NOTE: 'AAAAAA' REPRESENTS THE PRESENT ADDRESS

THE ADDRESS TYPED OUT MAY BE USED BY TYPING A 'CR' ONLY. A NEW ADDRESS IS ENTERED BY TYPING THE NEW ADDRESS AND TERMINATING THE ENTRY WITH A 'CR'. (THE NEW ADDRESS WILL BE USED UNTIL ANOTHER ADDRESS IS ENTERED.)

AFTER THE OPERATOR HAS RESPONDED TO THE ADDRESS ENTRY REQUEST, THE PROGRAM WILL VERIFY THAT THE RP11E ADDRESS RESPONDS. IF NO RESPONSE IS RECEIVED FROM THE RP11E ADDRESS, THE FOLLOWING ERROR MESSAGE IS TYPED:

RP11E DIDN'T RESPOND TO ADDRESSING
RPADR
AAAAA

THE PROGRAM RETURNS TO THE ADDRESS ENTRY ROUTINE. THE PROGRAM WILL NOT CONTINUE UNTIL THE OPERATOR ENTERS AN ADDRESS FOR WHICH THERE IS A UNIBUS RESPONSE.

4.5 CONVERSATION MODE

THE PROGRAM CONTAINS A 'CONVERSATION MODE' ROUTINE WHICH ALLOWS THE OPERATOR TO SELECT AND EXECUTE INDIVIDUAL TESTS AND TO SPECIFY THE DRIVE TO BE USED. THE CONVERSATION MODE ALSO ALLOWS THE OPERATOR TO CHANGE THE OPERATING PARAMETERS OF THE 'DATA STORAGE/RETRIEVAL TEST' (TEST 14).

WHEN THE PROGRAM ENTERS CONVERSATION MODE, THE FOLLOWING MESSAGE IS TYPED:

ENTER TEST #:

A 'CR' MAY BE ENTERED OR A VALID TEST NUMBER (E.G. 0 - 17) MAY BE ENTERED. THE DEFAULT ENTRY ('CR') WILL CAUSE THE PROGRAM TO SELECT TESTS 0 - 14. IF A SINGLE TEST NUMBER IS ENTERED, ONLY THAT TEST WILL BE EXECUTED.

THE PROGRAM WILL THEN ASK FOR A DRIVE ASSIGNMENT.

ENTER DRIVE #:

A 'CR' ENTRY WILL CAUSE THE PROGRAM TO SELECT ALL ONLINE DRIVES FOR TESTING. IF A VALID DRIVE NUMBER IS ENTERED (0 - 7) THE PROGRAM WILL CHECK THE STATE OF THE DRIVE: IF THE DRIVE IS NOT ONLINE THE DRIVE'S STATUS WILL BE TYPED AND THE PROGRAM WILL ASK FOR ANOTHER DRIVE ENTRY.

IF TEST 15, 16, OR 17 IS SELECTED, A 'CR' RESPONSE TO THE DRIVE NUMBER REQUEST WILL NOT BE ACCEPTED; A SPECIFIC DRIVE NUMBER MUST BE ENTERED FOR TEST 15, 16, OR 17.

IF TESTS 0 - 13, 15, 16 OR 17 HAVE BEEN SELECTED, OR IF THE ENTIRE TEST SEQUENCE WERE SPECIFIED, THE PROGRAM WILL BEGIN EXECUTING THE TEST OR SEQUENCE IMMEDIATELY. IF TEST 14 IS SPECIFIED, THE PROGRAM WILL ASK IF THE OPERATOR WANTS TO CHANGE PARAMETERS FOR TEST 14. THE FOLLOWING MESSAGE IS TYPED.

CHANGE PARAMETERS FOR TEST 14 ?

EITHER A 'CR' OR A 'Y' MAY BE ENTERED. IF A 'CR' IS ENTERED, THE PROGRAM WILL BYPASS THE TEST 14 PARAMETER ENTRY ROUTINE AND START TEST 14. IF A 'Y' IS ENTERED, THE PROGRAM STARTS THE TEST 14 PARAMETER CHANGE DIALOG.

THE PROGRAM WILL THEN TYPE THE WORD LENGTH FOR ALL TEST 14 TRANSFERS.

MAXIMUM WORD COUNT FOR TEST 14 (IN OCTAL): SSSSSS

NOTE: 'SSSSSS' = THE OCTAL VALUE OF THE WORD COUNT

THE PROGRAM WILL THEN ASK FOR A NEW WORD COUNT.

ENTER NEW WORD COUNT:

EITHER A VALUE (LESS THAN THE MAXIMUM TYPED ABOVE) OR A 'CR' MAY BE ENTERED. IF A VALUE IS ENTERED, THE NEW VALUE WILL BE USED FOR ALL DATA TRANSFER OPERATIONS IN TEST 14. WHEN THE PROGRAM IS RESTARTED OR A 'CONTROL C' IS TYPED, THE WORD COUNT VALUE WILL BE RETURNED TO THE SYSTEM DEFAULT VALUE. IF A 'CR' IS TYPED, THE PROGRAM WILL USE THE DEFAULT WORD COUNT VALUE.

THE PROGRAM WILL THEN ASK FOR A PATTERN SELECTION WORD ENTRY.

ENTER PATTERN SELECTION CODE FOR TEST 14:

ENTER THE OCTAL VALUE OF THE PATTERN SELECTION WORD OR A 'CR' FOR THE DEFAULT VALUE. REFER TO SECTION 7.9 FOR A DESCRIPTION OF THE PATTERN SELECTION WORD AND ITS USE IN TEST 14. IF TESTING IS TO BE AT A SPECIFIC DISK LOCATION, ENTER THE CODE FOR A SINGLE PATTERN. (IF TESTING IS AT A SPECIFIC ADDRESS, THE PROGRAM WILL NOT ROTATE THE DATA PATTERNS BUT WILL USE THE LOWEST NUMBERED PATTERN SPECIFIED BY THE PATTERN SELECT WORD.)

THE PROGRAM WILL ASK IF THE OPERATOR WANTS TO TEST AT A SPECIFIC DISK ADDRESS.

USE A SPECIFIC DISK ADDRESS FOR TEST 14 ?

ENTER EITHER A 'CR' OR A 'Y'. IF A 'Y' IS ENTERED, THE PROGRAM WILL TYPE THE FOLLOWING REMINDER MESSAGE ABOUT DATA PATTERN USAGE.

(A SINGLE DATA PATTERN MUST HAVE BEEN SPECIFIED
IF NOT, PATTERN 0 (ALL ZEROS) WILL BE USED)

THE PROGRAM WILL ASK FOR CYLINDER, TRACK, AND SECTOR ADDRESSES. AN ADDRESS MUST BE ENTERED. THE PROGRAM WILL NOT ACCEPT DEFAULT ENTRIES FOR THESE ADDRESSES. THE ENTRY REQUESTS FOR THE ADDRESSES ARE GIVEN BELOW.

ENTER CYLINDER ADDRESS:

ENTER TRACK ADDRESS:

ENTER SECTOR ADDRESS:

IF A SPECIFIC ADDRESS IS ENTERED FOR TESTING, TESTING WILL TAKE PLACE ONLY AT THAT ADDRESS WITH THE WORD COUNT AND PATTERN ENTERED ABOVE. THE TEST WILL MAKE ONE PASS AND GO TO END OF TEST. SW<14> SHOULD BE SET TO INHIBIT THE END OF TEST MESSAGE TYPEOUT.

TO TERMINATE THE TEST OR SEQUENCE, EITHER TYPE A 'CONTROL C' OR HALT THE PROCESSOR AND RESTART AT THE APPROPRIATE STARTING ADDRESS.

4.6 OPERATION OF TESTS 15, 16, & 17

TEST 15 AND TEST 17 CONTAIN IMBEDDED OPERATING INSTRUCTIONS. SETUP AND OPERATING INSTRUCTIONS ARE TYPED OUT AS THE TEST PROCEEDS.

TO OPERATE TEST 16, 'HEAD ALIGNMENT ROUTINE', REFER TO THE APPLICABLE MAINTENANCE MANUALS FOR THE DRIVE SETUP PROCEDURES. THE PROGRAM WILL TYPEOUT THE CONTROLLER SETUP PROCEDURES. TO CHANGE HEAD SELECTION, SW<07> MUST BE TOGGLED (TURNED ON, THEN OFF).

TO SETUP THE RP11E FOR HEAD ALIGNMENT, CONNECT A JUMPER BETWEEN PINS J13M2 AND J13C2 ON THE BACKPLANE.

5. ERROR HANDLING

5.1 ERROR REPORTING

WHEN THE PROGRAM DETECTS AN ERROR, THE ERROR ROUTINE IS CALLED; IF SW<13> IS NOT SET, THE ERROR MESSAGE FOR THAT ERROR WILL BE TYPED. EACH ERROR TYPEOUT CONTAINS THE FOLLOWING:

- A. AN ERROR MESSAGE
- B. A DATA HEADER LINE
- C. A DATA LINE CONTAINING
 1. THE TEST NUMBER
 2. THE ADDRESS (PROGRAM COUNTER CONTENTS) WHERE THE ERROR CALL WAS MADE
 3. CONTENTS OF THE APPROPRIATE REGISTERS
 4. OTHER SPECIFIC DATA

5.2 ERROR TYPES

THE ERRORS DETECTED ARE DIVIDED INTO 2 CATEGORIES: FATAL AND NON-FATAL.

FATAL ERRORS ARE DRIVE UNSAFE ERRORS WHICH OCCUR ON THE DRIVE BEING USED FOR TESTING OR DRIVES WHICH GO OFFLINE DURING TESTING. IF EITHER OF THESE TWO CONDITIONS OCCUR, THE PROGRAM WILL TERMINATE TESTING USING THE DRIVE WHICH GAVE THE FATAL ERROR.

ALL OTHER ERROR TYPES ARE NON-FATAL. E.G., TESTING CONTINUES AFTER THE ERROR IS REPORTED.

6. RESTRICTIONS

6.1 SUBSYSTEMS WITH MIXED DRIVE TYPES

THE PROGRAM ASSUMES THAT ONLY ONE KIND OF DRIVE IS ATTACHED TO A SYSTEM (E.G. RPO2'S & RPRO2'S OR RPO3'S). IF BOTH RPO2'S (AND RPRO2'S) OR RPO3'S ARE ON THE SAME SYSTEM, THE PROGRAM WILL ASSUME THAT ALL OF THE DRIVES ARE RPO3'S. TO TEST DRIVES ON A MIXED SYSTEM, TEST ALL OF THE SAME KIND OF DRIVE AT ONE TIME. THE OTHER DRIVES ON THE CONTROLLER MUST BE POWERED DOWN.

6.2 FORMATTED PACKS

EACH OF THE DRIVES BEING USED MUST HAVE A FORMATTED DISK PACK.

6.3 'CONTROL C' TERMINATION DURING TESTS 5 & 13

IF TESTS 5 OR 13 ARE TERMINATED WITH A 'CONTROL C', CYLINDER 0, TRACK 0, SECTOR 0 MAY BE LEFT WITH EITHER AN INVALID HEADER OR AN INVALID DATA FIELD. IF TEST 5 IS STOPPED, THE HEADER FOR SECTOR 0 MAY STILL BE MISFORMATTED; IF TEST 13 IS STOPPED, SECTOR 0 MAY BE STILL RECORDED IN 10/15 MODE.

6.4 CONTROLLER INTERRUPT

THE PROGRAM DOES NOT USE CONTROLLER INTERRUPT. THE PROGRAM WAITS FOR THE CONTROLLER 'READY' OR DRIVE 'READY' BITS TO SET TO INDICATE ORDER TERMINATION.

7. MISCELLANEOUS

7.1 RUN TIME

TO MAKE ONE PASS OF THE PROGRAM (TESTS 0 - 14) TAKES APPROXIMATELY
20 MINUTES FOR EACH RPO2 OR RPRO2 USED AND APPROXIMATELY 40 MINUTES
FOR EACH RPO3 USED.

7.2 STACK POINTER

STACK IS INITIALLY SET TO 1100 AND EXTENDS DOWNWARD IN MEMORY.

7.3 END OF PASS/END OF TEST

THE PROGRAM WILL PERFORM ONE PASS FOR EACH DRIVE BEING USED FOR
TESTING. END OF TEST WILL OCCUR WHEN EACH DRIVE BEING USED FOR
TESTING HAS HAD ONE PASS.

7.4 SUBROUTINE CALLS

THE SUBROUTINE CALLS USED BY THE PROGRAM ARE:

- A. 'SCOPE' (IOT INSTRUCTION). THIS CALL IS PLACED BETWEEN EACH
TEST IN THE PROGRAM. THIS ROUTINE ESTABLISHES THE TEST ITERATION
COUNT AND LOOP ON TEST AND LOOP ON ERROR ADDRESSES.
- B. 'ERROR' (EMT INSTRUCTION). THIS CALL IS USED TO REPORT ALL
ERRORS. A CALL TO THIS ROUTINE IS FOLLOWED BY A NUMBER WHICH
IDENTIFIES THE ERROR MESSAGE WHICH WILL BE TYPED.

THE TRAP INSTRUCTION IS USED FOR THE FOLLOWING SUBROUTINE CALLS:

TYPE - TTY TYPEOUT ROUTINE
TYPOC - TYPE OCTAL NUMBER (WITH LEADING ZEROS)
TYPOS - TYPE OCTAL NUMBER (NO LEADING ZEROS)
TYPDS - TYPE DECIMAL NUMBER
RDCHR - READ A CHARACTER FROM THE TTY KEYBOARD
RDLIN - READ A LINE FROM THE TTY KEYBOARD
SAVREG - ROUTINE TO SAVE R0 - R5
RESREG - ROUTINE TO RESTORE R0 - R5

7.5 DISK SURFACE USAGE

THE PROGRAM USES THE FOLLOWING DISK LOCATIONS FOR TESTING.

<u>TEST</u>	<u>DISK ADDRESS</u>
0	CYL 0, TRK 0, SEC 0
1	CYL 0, TRK 0, SEC 0
2	CYL 0, TRK 0, SEC 0
3	CYL 0, TRK 0, SEC 0 MAXIMUM CYL, TRK 19, SEC 9
4	CYL 0, TRK 0, SEC 0
5	CYL 0, TRK 0, SEC 0
6	CYL 192, TRK 0, SEC 0
7	CYL 0, TRK 0, SEC 0
10	CYL 0, TRK 0, SEC 0 - 9
11	CYL 0, TRK 0 - 19, SEC 0
12	CYL 0, TRK 0, SEC 0
13	CYL 0, TRK 0, SEC 0
14	ENTIRE PACK
15	MAXIMUM CYL, TRK 0, SEC 0
16	N/A
17	NONE

7.6 TEST EXECUTION SEQUENCE

THE TEST SEQUENCE IS TESTS 0 - 14. TESTS 15, 16, & 17 ARE ONLY PERFORMED IF DIRECTED BY THE OPERATOR FROM CONVERSATION MODE. TESTS 15, 16, OR 17 CANNOT BE EXECUTED IN SEQUENCE.

7.7 DRIVE STATUS TYPEOUT

WHEN THE PROGRAM IS STARTED FROM LOCATION 200, 204, 210, OR 214 OR ENTERS THE CONVERSATION ROUTINE FROM A 'CONTROL C', THE PROGRAM WILL TYPE THE STATUS OF DRIVES 0 - 7. THIS DRIVE STATUS TYPEOUT CAN BE INHIBITED BY SETTING SW<07> BEFORE THE PROGRAM IS STARTED OR A 'CONTROL C' IS TYPED.

7.8 DATA COMPARISON USING MEMORY MANAGEMENT

IF THE SYSTEM HAS MORE THAN 28K, TEST 14 WILL ROTATE THE READ BUFFERS THROUGH ALL OF MEMORY. IF A DATA COMPARISON ERROR IS DETECTED, THE PROGRAM WILL TYPEOUT THE 'GOOD' AND THE 'BAD' DATA AND THE V I R T U A L ADDRESS OF THE COMPARISON ERROR LOCATION. THE CONTENTS OF THE APRA'S IN USE WILL ALSO BE TYPED. IF MEMORY MANAGEMENT IS ENABLED, THE PROGRAM PERFORMS ALL DATA COMPARISONS FROM VIRTUAL LOCATIONS 60000 - 137776 USING APRA'S 3, 4, & 5. THE PHYSICAL ADDRESS OF THE COMPARISON ERROR CAN BE FORCED BY ADDING THE APPLICABLE 'KIPRA' REGISTER CONTENTS (THE REGISTERS ARE DISPLAYED TO THE VIRTUAL ADDRESS OF THE ERROR.

7.9 PATTERN SELECTION WORD

THE DATA PATTERNS USED BY TEST 14 ARE SELECTED FROM A SELECTION WORD WHICH IS LOADED BY THE PROGRAM OR LOADED BY THE OPERATOR IN CONVERSATION MODE. THE PATTERN SELECTION WORD CONTAINS A BIT FOR EACH PATTERN TO BE USED; BIT00 REPRESENTS PATTERN 0, BIT01 REPRESENTS PATTERN 1, ETC. REFER TO SECTION 8.13 FOR THE CONTENTS OF EACH PATTERN.

THE OCTAL VALUES FOR EACH DATA PATTERN ARE AS FOLLOWS.

PATTERN #	OCTAL SELECTION CODE
0	100000
1	200000
2	300000
3	400000
4	500000
5	600000
6	700000
7	100000
8	200000
9	300000
10	400000
11	500000
12	600000
13	700000
14	100000
15	200000

THE INDIVIDUAL OCTAL CODES ARE 'OR ED' TOGETHER TO FORM THE PATTERN SELECTION WORD. TO SELECT PATTERNS 0, 6, & 13, THE PATTERN SELECTION WORD COULD BE THE FOLLOWING OCTAL VALUE: 20101.

7.10 CONTROLLER/DRIVE OPERATION TIME LIMIT

ALL CONTROLLER AND DRIVE OPERATIONS ARE TIMED BY THE PROGRAM. IF AN OPERATION EXCEEDS THE TIME LIMIT, THE PROGRAM WILL REPORT THE TIMEOUT, RESET THE CONTROLLER, TERMINATE THE CURRENT TEST, AND GO TO THE NEXT TEST. THE TIME LIMIT IS 150 MS TO 1.5 SEC (DEPENDING ON THE PROCESSOR

B. TEST DESCRIPTIONS

B.1 TEST 0 - 'CLEAR' COMMAND TERMINATION TEST

VERIFY THAT 'CLEAR' TERMINATES AN OPERATION AND CAUSES CONTROLLER READY TO SET.

B.2 TEST 1 - WRITE CHECK COMMAND TEST

THIS TEST VERIFIES THE WRITE CHECK LOGIC AND TO VERIFY THAT A WRITE CHECK ERROR CAN BE DETECTED. 'FLOATING ONES' AND 'FLOATING ZEROS' PATTERNS ARE USED TO TEST THE WRITE CHECK COMPARE LOGIC.

B.3 TEST 2 - PARTIAL SECTOR WRITE TEST

CHECK THE ABILITY OF THE RP11E TO CLEAR THE REMAINDER OF A SECTOR ON A PARTIAL WRITE OPERATION. A SECTOR OF ALL ONES IS WRITTEN AND THEN A TWO WORD WRITE OPERATION IS PERFORMED. THE SECTOR IS THEN READ BACK AND VERIFIED. THE FIRST TWO WORDS SHOULD BE ONES AND THE REST SHOULD BE ZEROS.

B.4 TEST 3 - 'EOP' TEST

TEST THAT 'EOP' SETS WHEN THE CONTROLLER TRIES TO WRITE BEYOND THE LIMITS OF THE DRIVE. THE FIRST SECTOR OF THE PACK IS WRITTEN WITH ZEROS. THEN A TWO SECTOR WRITE OF ALL ONE'S IS ISSUED FOR THE MAXIMUM BE SET. THE FIRST SECTOR OF THE PACK IS CHECKED TO MAKE SURE THAT IS STILL CONTAINS ZEROS.

B.5 TEST 4 - 'PROG' ERROR TEST

VERIFY THAT THE RP11E GENERATES A 'PROG' ERROR IF A COMMAND IS ISSUED WHILE THE CONTROLLER IS BUSY.

B.6 TEST 5 - 'HEADER NOT FOUND' TEST

MISFORMAT THE FIRST SECTOR ON THE PACK AND THEN READ IT BACK. VERIFY THAT READ AND WRITE HEADER OPERATIONS WILL TRANSFER DATA CORRECTLY. ISSUE A WRITE COMMAND TO SECTOR ZERO. THIS SHOULD CAUSE 'HEADER NOT FOUND' TO SET. THE TEST SECTOR IS REFORMATTED.

B.7 TEST 6 - COMMAND BUFFERING TEST

ISSUE A SEEK COMMAND AND WAIT FOR DONE TO SET. THEN ISSUE A READ COMMAND WHILE THE HEADS ARE STILL MOVING. THE RP11E SHOULD HOLD THE READ COMMAND UNTIL THE SEEK IS COMPLETE.

B.8 TEST 7 - 'NXME' ERROR TEST

THE NON-EXISTENT MEMORY ERROR BIT IS TESTED BY ATTEMPTING TO READ 1 WORD INTO LOCATION 76000. 'NXME', 'HE', & 'ERR' SHOULD ALL BE SET.

8.9 TEST 10 - SECTOR ADDRESSING TEST

THE SECTOR ADDRESSING TEST TESTS THE CAPABILITY OF THE RP11E TO LOCATE A SECTOR BY USING THE 'SOT' COUNTER (FOR 'HOR' OPERATIONS) AND TO LOCATE A SECTOR BY COMPARING THE HEADER CONTENTS AGAINST THE CYLINDER AND TRACK SECTOR ADDRESS REGISTERS (NORMAL MODE). THE TEST IS PERFORMED IN 3 PARTS

1. WRITE ALL HEADERS ON TRACK 0, CYLINDER 0 IN ASCENDING SEQUENCE FROM INDEX (0, 1, 2 ... 9).

2. READ EACH HEADER, BEGINNING WITH SECTOR 0, AND VERIFY THAT THE SECTOR FIELD IN THE HEADER IS CORRECT FOR THE HEADER EXPECTED.

VERIFY THAT THE 'SOT' COUNTER AND THE SECTOR ADDRESS IN 'RPDA' ARE EQUAL AND ARE THE EXPECTED VALUE. THE EXPECTED VALUE IS 1 GREATER THAN THE SECTOR READ. THE SECTOR ADDRESS REGISTER AND THE 'SOT' WILL BE INCREMENTED BY THE TIME THE PROGRAM IS ABLE TO READ THE REGISTER.

WRITE THE SECTOR'S ADDRESS IN THE DATA FIELD. (FOR SECTOR 0, AN OCTAL 12 IS WRITTEN.)

CONTINUE THIS SEQUENCE FOR THE REMAINING SECTORS ON THE TRACK.

3. READ THE DATA FROM EACH SECTOR USING A NORMAL (I.E. NON-HEADER) READ AND VERIFY THAT THE DATA IS CORRECT FOR THE EXPECTED SECTOR.

8.10 TEST 11 - TRACK ADDRESSING TEST

TRACK ADDRESSING IS TESTED BY WRITING THE TRACK'S ADDRESS AS DATA IN SECTOR 0 OF EACH TRACK IN CYLINDER 0. SECTOR 0 IS THEN READ BACK AND THE DATA IS CHECKED TO VERIFY THAT THE PROPER HEAD WAS SELECTED.

8.11 TEST 12 - EXTENDED MEMORY ADDRESS BIT TEST

THIS TEST TESTS THE OPERATION OF THE EXTENDED MEMORY ADDRESS BITS. IF THE SYSTEM DOES NOT HAVE MEMORY MANAGEMENT OR HAS MEMORY MANAGEMENT AND ONLY 32K, THE TEST WILL NOT BE PERFORMED.

1. THE PROGRAM WRITES A 2 WORD TEST SECTOR OF ALL ONES.

2. EXTENDED ADDRESS BIT 'MEX00' IS TESTED BY CLEARING LOCATION 200000 AND READING THE TEST SECTOR INTO LOCATION 177776. LOCATION 200000 IS CHECKED TO VERIFY THAT THE DATA IS CORRECT (ONES). THE PROGRAM VERIFIES THAT 'MEX00' HAS SET AND THAT 'MEX01' IS NOT SET.

3. IF THE SYSTEM HAS AT LEAST 64K, 'MEX01' IS TESTED. LOCATION 400000 IS CLEARED AND THE TEST SECTOR IS READ INTO LOCATION 377776 ('MEX00' IS SET FOR THE READ). LOCATION 400000 IS CHECKED FOR THE PROPER CONTENTS (ONES). 'MEX00' SHOULD HAVE RESET AND 'MEX01' SHOULD BE SET.

8.12 TEST 13 - RP11E DATA BUFFER REGISTER BIT TEST

THE DATA BUFFER REGISTER TEST VERIFIES THAT ALL 36 BITS OF THE DATA BUFFER REGISTER CAN BE SET AND CLEARED.

1. THE TEST WRITES THE FOLLOWING TEST PATTERN IN CYLINDER 0, TRACK 0, SECTOR 0 USING 10/15 MODE ('MODE' BIT SET).

052525
052525
000005
125252
125252
125252
000012
052525
052525
000005
125252

ETC

2. THE TEST SECTOR IS THEN READ AND THE DATA IS COMPARED. AT THE COMPLETION OF THE TEST, THE PROGRAM RESTORES THE TEST SECTOR TO PDP-11 MODE.

8.13 TEST 14 - DATA STORAGE/RETRIEVAL TEST

THE DATA TEST PERFORMS DATA STORAGE AND RETREIVAL TESTING USING THE CONTROLLER AND THE CURRENTLY SELECTED DRIVE. UNLESS ALTERED BY THE OPERATOR, THE TEST WILL WRITE AND WRITE CHECK THE ENTIRE DISK PACK USING ALL 16 DATA PATTERNS IN A ROTATING MANNER. THE BUFFER SIZE USED FOR THE WRITE AND WRITE CHECK ORDERS IS DETERMINED BY THE AVAILABLE MEMORY ON THE SYSTEM (MINUS THE SPACE REQUIRED BY THE PROGRAM LOADER). THE PROGRAM WILL USE THE AVAILABLE MEMORY SIZE OR 8K, WHICHEVER IS LESS, AS THE BUFFER SIZE.

AFTER THE PACK HAS BEEN WRITTEN, THE TEST WILL READ AND COMPARE THE DATA. THE BUFFER SIZE USED FOR THE READ IS THE SAME SIZE AS THAT USED FOR THE WRITE/WRITE CHECK SEGEMENT OF THE TEST. FOR THE READ SEGEMENT OF THE TEST, THE PROGRAM ROTATES THE READ BUFFER THROUGH MEMORY SO THAT ALL OF AVAILABLE MEMORY IS USED BY THE READ AND COMPARE OPERATIONS. THE TEST USES MEMORY MANAGEMENT ON SYSTEMS WITH MORE THAN 28K. (MEMORY MANAGEMENT WILL NOT BE USED IF THE TEST IS RUN ON MEMORY MANAGEMENT SYSTEMS WITH 28K OR LESS.)

IF AN ERROR OCCURS DURING A READ OPERATION (EXCEPT DATA COMPARSION ERRORS), THE PROGRAM WILL RETRY THE READ ORDER UP TO 3 TIMES. WRITE AND WRITE CHECK ORDERS ARE NOT RETRIED.

THE DATA PATTERNS USED BY THE DATA TEST ARE GIVEN BELOW:

PAT 0	PAT 1	PAT 2	PAT 3	PAT 4	PAT 5	PAT 6	PAT 7
000000	000001	177776	000000	000000	052323	007417	026455
000000	000003	177774	000000	010421	052323	007417	026455
000000	000007	177770	000000	021042	052323	007417	026455
000000	000017	177760	177777	031463	052323	170360	113233
000000	000037	177740	177777	042104	052323	170360	113233
000000	000077	177700	177777	052525	052323	170360	113233
000000	000177	177600	000000	063146	052323	007417	026455
000000	000377	177400	000000	073567	052323	007417	026455
000000	000777	177000	177777	084210	052323	170360	113233
000000	001777	176000	177777	114463	052323	170360	113233
000000	003777	174000	000000	125104	052323	007417	026455
000000	007777	170000	177777	135673	052323	170360	113233
000000	017777	160000	000000	146314	052323	007417	026455
000000	037777	140000	177777	156735	052323	170360	113233
000000	077777	100000	177777	167356	052323	170360	113233
000000	177777	000000	000000	177777	052323	007417	026455
PAT 8	PAT 9	PAT 10	PAT 11	PAT 12	PAT 13	PAT 14	PAT 15
077577	000001	177776	172666	077777	177777	000000	177777
077577	000002	177775	155555	137777	177777	000000	177777
077577	000004	177773	172666	157777	177777	177777	000000
077577	000010	177767	155555	167777	177777	177777	000000
077577	000020	177757	172666	173777	177777	177777	000000
077577	000040	177737	155555	175777	177777	177777	000000
077577	000100	177677	172666	176777	177777	177777	000000
077577	000200	177577	155555	177377	177777	177777	000000
077577	000400	177377	172666	177577	177777	177777	000000
077577	001000	176777	155555	177677	177777	177777	000000
077577	002000	175777	172666	177737	177777	177777	000000
077577	004000	173777	155555	177757	177777	177777	000000
077577	010000	167777	172666	177767	177777	177777	000000
077577	020000	157777	155555	177773	177777	177777	000000
077577	040000	137777	172666	177775	177777	177777	000000
077577	100000	077777	155555	177776	177777	177777	000000

E.14 TEST 15 - POWER FAIL TEST

TEST THE ABILITY OF THE RP11E TO SENSE POWER FAILURE AND FOR THE DRIVES TO RETRACT THEIR HEADS. WHEN POWER IS TURNED ON AGAIN THE CYLINDER ADDRESS IS TESTED FOR ZERO. AFTER TYPING THE MESSAGE REQUESTING POWER TO BE TURNED OFF, THE PROGRAM GOES INTO A LOOP READING FROM THE SELECTED DISK DRIVE. AFTER POWER IS RESTORED, MEMORY IS CHECKED TO SEE THAT THE DISK DID NOT TRANSFER ANY DATA TO MEMORY WHILE POWER WAS GOING DOWN.

E.15 TEST 16 - HEAD ALIGNMENT ROUTINE

THIS ROUTINE PERFORMS HEAD SELECTION AS DIRECTED FROM THE KEYBOARD; ALIGNMENT OF THE SELECTED HEAD MAY BE CHECKED OR ADJUSTED.

8.16 TEST 17 - RPIIE DRIVE CONTROL PANEL SWITCH TEST

THIS TEST TESTS THE 'ENABLE/DISABLE' AND 'READ ONLY' SWITCHES ON THE DRIVE CONTROL PANEL AND TESTS THE 'WRITE LOCKOUT' AND 'LOA' SWITCHES ON THE RPIIE CONTROL PANEL.

9. PROGRAM LISTING

967
966
965
964
963
962
961
960
959
958
957
956
955
954
953
952
951
950
949
948
947
946
945
944
943
942
941
940
939
938
937
936
935
934
933
932
931
930
929
928
927
926
925
924
923
922
921
920
919
918
917
916
915
914
913
912
911
910
909
908
907
906
905
904
903
902
901
900
899
898
897
896
895
894
893
892
891
890
889
888
887
886
885
884
883
882
881
880
879
878
877
876
875
874
873
872
871
870
869
868
867
866
865
864
863
862
861
860
859
858
857
856
855
854
853
852
851
850
849
848
847
846
845
844
843
842
841
840
839
838
837
836
835
834
833
832
831
830
829
828
827
826
825
824
823
822
821
820
819
818
817
816
815
814
813
812
811
810
809
808
807
806
805
804
803
802
801
800
799
798
797
796
795
794
793
792
791
790
789
788
787
786
785
784
783
782
781
780
779
778
777
776
775
774
773
772
771
770
769
768
767
766
765
764
763
762
761
760
759
758
757
756
755
754
753
752
751
750
749
748
747
746
745
744
743
742
741
740
739
738
737
736
735
734
733
732
731
730
729
728
727
726
725
724
723
722
721
720
719
718
717
716
715
714
713
712
711
710
709
708
707
706
705
704
703
702
701
700
699
698
697
696
695
694
693
692
691
690
689
688
687
686
685
684
683
682
681
680
679
678
677
676
675
674
673
672
671
670
669
668
667
666
665
664
663
662
661
660
659
658
657
656
655
654
653
652
651
650
649
648
647
646
645
644
643
642
641
640
639
638
637
636
635
634
633
632
631
630
629
628
627
626
625
624
623
622
621
620
619
618
617
616
615
614
613
612
611
610
609
608
607
606
605
604
603
602
601
600
599
598
597
596
595
594
593
592
591
590
589
588
587
586
585
584
583
582
581
580
579
578
577
576
575
574
573
572
571
570
569
568
567
566
565
564
563
562
561
560
559
558
557
556
555
554
553
552
551
550
549
548
547
546
545
544
543
542
541
540
539
538
537
536
535
534
533
532
531
530
529
528
527
526
525
524
523
522
521
520
519
518
517
516
515
514
513
512
511
510
509
508
507
506
505
504
503
502
501
500
499
498
497
496
495
494
493
492
491
490
489
488
487
486
485
484
483
482
481
480
479
478
477
476
475
474
473
472
471
470
469
468
467
466
465
464
463
462
461
460
459
458
457
456
455
454
453
452
451
450
449
448
447
446
445
444
443
442
441
440
439
438
437
436
435
434
433
432
431
430
429
428
427
426
425
424
423
422
421
420
419
418
417
416
415
414
413
412
411
410
409
408
407
406
405
404
403
402
401
400
399
398
397
396
395
394
393
392
391
390
389
388
387
386
385
384
383
382
381
380
379
378
377
376
375
374
373
372
371
370
369
368
367
366
365
364
363
362
361
360
359
358
357
356
355
354
353
352
351
350
349
348
347
346
345
344
343
342
341
340
339
338
337
336
335
334
333
332
331
330
329
328
327
326
325
324
323
322
321
320
319
318
317
316
315
314
313
312
311
310
309
308
307
306
305
304
303
302
301
300
299
298
297
296
295
294
293
292
291
290
289
288
287
286
285
284
283
282
281
280
279
278
277
276
275
274
273
272
271
270
269
268
267
266
265
264
263
262
261
260
259
258
257
256
255
254
253
252
251
250
249
248
247
246
245
244
243
242
241
240
239
238
237
236
235
234
233
232
231
230
229
228
227
226
225
224
223
222
221
220
219
218
217
216
215
214
213
212
211
210
209
208
207
206
205
204
203
202
201
200
199
198
197
196
195
194
193
192
191
190
189
188
187
186
185
184
183
182
181
180
179
178
177
176
175
174
173
172
171
170
169
168
167
166
165
164
163
162
161
160
159
158
157
156
155
154
153
152
151
150
149
148
147
146
145
144
143
142
141
140
139
138
137
136
135
134
133
132
131
130
129
128
127
126
125
124
123
122
121
120
119
118
117
116
115
114
113
112
111
110
109
108
107
106
105
104
103
102
101
100
99
98
97
96
95
94
93
92
91
90
89
88
87
86
85
84
83
82
81
80
79
78
77
76
75
74
73
72
71
70
69
68
67
66
65
64
63
62
61
60
59
58
57
56
55
54
53
52
51
50
49
48
47
46
45
44
43
42
41
40
39
38
37
36
35
34
33
32
31
30
29
28
27
26
25
24
23
22
21
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0

```

TITLE MD-11-DZRPY-C, RP11-E FUNCTIONAL LOGIC & R/W TEST
*COPYRIGHT (C) 1975, 1977
*DIGITAL EQUIPMENT CORP.
*MAYNARD, MASS. 01754
*
*PROGRAM BY C. HESS/F. ROEMER
*
*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
*
SBTTL OPERATIONAL SWITCH SETTINGS
*
SWITCH          USE
-----          -----
          15     HALT ON ERROR
          14     LOOP ON TEST
          13     INHIBIT ERROR TYPEOUTS
          11     INHIBIT ITERATIONS
          10     BELL ON ERROR
           9     LOOP ON ERROR
           8     DON'T TYPE DRIVE STATUS WHEN PROGRAM STARTED OR WHEN
                ENTERING CONVERSATION MODE
           7     CHANGE HEAD SELECTION IN TEST 16
           6     DON'T USE MEMORY MANAGEMENT IN TEST 14
           5     DISPLAY ALL READ RETRY ATTEMPTS IN TEST 14
           4     PRINT ALL COMPARE ERRORS IN TESTS 13 & 14
           3     INHIBIT DATA COMPARE IN TESTS 13 & 14
           2     BYPASS READ AND DATA COMPARE IN TEST 14
           1     BYPASS WRITE CHECK IN TEST 14
           0     BYPASS WRITE IN TEST 14

```

.SBTTL TRAP CATCHER

00000C

```

.=0
*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2.HALT"
*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS

```

```

000174 000174
000176 000000

```

```

.=174
DISPREG: .WORD 0      ;; SOFTWARE DISPLAY REGISTER
SWREG:   .WORD 0      ;; SOFTWARE SWITCH REGISTER

```

.SBTTL ACT11 HOOKS

```

000046 00020C
000052 000046
000052 017050
000200 000052
000200 00000C
000200 000200

```

```

*****
HOOKS REQUIRED BY ACT11
$SVPC=.          ;SAVE PC
.=46             ;;1)SET LOC.46 TO ADDRESS OF SENDAC IN .SECF
SENDAC           ;;
.=52             ;;2)SET LOC.52 TO ZERO
.WORD 0          ;;
.$SVPC           ;; RESTORE PC

```

.SBTTL STARTING ADDRESSES

00020C

.=20C

STARTING ADDRESSES

9777
9778 000200 000137 003654
9779
980 000204 000137 003676
981
982 000210 000137 003644
983
984 000214 000137 003666
985

:*200 = NORMAL START
JMP START1
:*204 = SELECT OPERATING PARAMETERS
JMP START2
:*210 = SELECT RP11E ADDRESS
JMP START3
:*214 = COMBINATION OF 204 AND 210
JMP START4

.SBTTL BASIC DEFINITIONS

986
987
988
989 001100

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

:*MISCELLANEOUS DEFINITIONS

993 000011
994 000012
995 000015
996 000200
997 177776
998
999 177774
1000
1001 177772
1002 177570
1003 177570
1004

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

:*GENERAL PURPOSE REGISTER DEFINITIONS

1005 000000
1006 000001
1007 000002
1008 000003
1009 000004
1010 000005
1011 000006
1012 000007
1013 000006
1014 000007
1015 000007
1016

R0= %0 ;:GENERAL REGISTER
R1= %1 ;:GENERAL REGISTER
R2= %2 ;:GENERAL REGISTER
R3= %3 ;:GENERAL REGISTER
R4= %4 ;:GENERAL REGISTER
R5= %5 ;:GENERAL REGISTER
R6= %6 ;:GENERAL REGISTER
R7= %7 ;:GENERAL REGISTER
SP= %6 ;:STACK POINTER
PC= %7 ;:PROGRAM COUNTER

:*PRIORITY LEVEL DEFINITIONS

1017 000000
1018 000040
1019 000100
1020 000140
1021 000200
1022 000240
1023 000300
1024 000340
1025
1026

PR0= 0 ;:PRIORITY LEVEL 0
PR1= 40 ;:PRIORITY LEVEL 1
PR2= 100 ;:PRIORITY LEVEL 2
PR3= 140 ;:PRIORITY LEVEL 3
PR4= 200 ;:PRIORITY LEVEL 4
PR5= 240 ;:PRIORITY LEVEL 5
PR6= 300 ;:PRIORITY LEVEL 6
PR7= 340 ;:PRIORITY LEVEL 7

:*"SWITCH REGISTER" SWITCH DEFINITIONS

1027 100000
1028 040000
1029 020000
1030 010000
1031 004000
1032

SW15= 100000
SW14= 40000
SW13= 20000
SW12= 10000
SW11= 4000

BASIC DEFINITIONS

1033 002000
1034 001000
1035 000400
1036 000200
1037 000100
1038 000040
1039 000020
1040 000010
1041 000004
1042 000002
1043 000001

SW10= 2000
SW09= 1000
SW08= 400
SW07= 200
SW06= 100
SW05= 40
SW04= 20
SW03= 10
SW02= 4
SW01= 2
SW00= 1
.EQUIV SW09, SW9
.EQUIV SW08, SW8
.EQUIV SW07, SW7
.EQUIV SW06, SW6
.EQUIV SW05, SW5
.EQUIV SW04, SW4
.EQUIV SW03, SW3
.EQUIV SW02, SW2
.EQUIV SW01, SW1
.EQUIV SW00, SW0

1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054

.*DATA BIT DEFINITIONS (BIT00 TO BIT15)

1055 100000
1056 040000
1057 020000
1058 010000
1059 004000
1060 002000
1061 001000
1062 000400
1063 000200
1064 000100
1065 000040
1066 000020
1067 000010
1068 000004
1069 000002
1070 000001
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082

BIT15= 100000
BIT14= 40000
BIT13= 20000
BIT12= 10000
BIT11= 4000
BIT10= 2000
BIT09= 1000
BIT08= 400
BIT07= 200
BIT06= 100
BIT05= 40
BIT04= 20
BIT03= 10
BIT02= 4
BIT01= 2
BIT00= 1
.EQUIV BIT09, BIT9
.EQUIV BIT08, BIT8
.EQUIV BIT07, BIT7
.EQUIV BIT06, BIT6
.EQUIV BIT05, BIT5
.EQUIV BIT04, BIT4
.EQUIV BIT03, BIT3
.EQUIV BIT02, BIT2
.EQUIV BIT01, BIT1
.EQUIV BIT00, BIT0

1083
1084 000004
1085 000010
1086 000014
1087 000014
1088 000014

.*BASIC "CPU" TRAP VECTOR ADDRESSES

ERRVEC= 4 ;: TIME OUT AND OTHER ERRORS
RESVEC= 10 ;: RESERVED AND ILLEGAL INSTRUCTIONS
TRIVVEC= 14 ;: "T" BIT
TRYVEC= 14 ;: TRACE TRAP
BFTVEC= 14 ;: BREAKPOINT TRAP (BPT)

BASIC DEFINITIONS

```

1089      000020      IOTVEC= 20      : INPUT/OUTPUT TRAP (IOT) **SCOPE**
1090      000024      PWRVEC= 24      : POWER FAIL
1091      000030      EMTVEC= 30      : EMULATOR TRAP (EMT) **ERROR**
1092      000034      TRAPVEC=34      : "TRAP" TRAP
1093      000060      TKVEC= 60      : TTY KEYBOARD VECTOR
1094      000064      TPVEC= 64      : TTY PRINTER VECTOR
1095      000240      PIRQVEC=240    : PROGRAM INTERRUPT REQUEST VECTOR
1096      .SBTTL      MEMORY MANAGEMENT DEFINITIONS
1097
1098      : *KT11 VECTOR ADDRESS
1099
1100      000250      MMVEC= 250
1101
1102      : *KT11 STATUS REGISTER ADDRESSES
1103
1104      177572      SR0= 177572
1105      177574      SR1= 177574
1106      177576      SR2= 177576
1107      172516      SR3= 172516
1108
1109      ; *KERNEL "I" PAGE DESCRIPTOR REGISTERS
1110
1111      172300      KIPDR0= 172300
1112      172302      KIPDR1= 172302
1113      172304      KIPDR2= 172304
1114      172306      KIPDR3= 172306
1115      172310      KIPDR4= 172310
1116      172312      KIPDR5= 172312
1117      172314      KIPDR6= 172314
1118      172316      KIPDR7= 172316
1119
1120      ; *KERNEL "I" PAGE ADDRESS REGISTERS
1121
1122      172340      KIPAR0= 172340
1123      172342      KIPAR1= 172342
1124      172344      KIPAR2= 172344
1125      172346      KIPAR3= 172346
1126      172350      KIPAR4= 172350
1127      172352      KIPAR5= 172352
1128      172354      KIPAR6= 172354
1129      172356      KIPAR7= 172356
1130
1131
1132      ; RP11E REGISTER INDEX EQUATES
1133
1134      000000      RPDS=00
1135      000002      RPER=02
1136      000004      RPCS=04
1137      000006      RPWC=06
1138      000010      RPBA=10
1139      000012      RPCA=12
1140      000014      RPOA=14
1141      000016      RPM1=16
1142      000020      RPM2=20
1143      000022      RPM3=22
1144      000024      SUCA=24

```

```

1145      000026      SILO=26
1146
1147      ;RP11 OP CODE DEFINITIONS
1148
1149      000001      CLEAR=1      ;CLEAR THE CONTROLLER
1150      000003      WRITSEK=3    ;WRITE WITH IMPLIED SEEK AND HEAD SELECTION
1151      000005      RDSEK=5      ;READ WITH IMPLIED SEEK AND HEAD SELECT
1152      000007      WRITCHK=7   ;WRITE CHECK WITH IMPLIED SEEK AND HEAD SELECT
1153      000011      SEEK=11     ;SEEK
1154      000013      WRITE=13    ;WRITE (NO IMPLIED SEEK OR HEAD SELECT)
1155      000015      HOMSEK=15   ;HOME SEEK
1156      000017      READ=17    ;READ (NO IMPLIED SEEK OR HEAD SELECT)
1157
1158
1159      ;RPDS REGISTER BIT DEFINITIONS
1160
1161      000400      SUWP=400     ;SELECTED UNIT WRITE PROTECTED
1162      001000      SUFU=1000   ;SELECTED UNIT FILE UNSAFE
1163      004000      SUSI=4000   ;SELECTED UNIT SEEK INCOMPLETE
1164      020000      SUP03=20000 ;SELECTED UNIT IS AN RP03
1165      040000      SUOL=40000  ;SELECTED UNIT IS ONLINE
1166      100000      SURDY=100000;SELECTED UNIT IS READY
1167
1168      ;RPCS REGISTER BIT DEFINITIONS
1169
1170      000020      MEX0=20     ;EXTENDED MEMORY BIT #1
1171      000040      MEX1=40     ;EXTENDED MEMORY BIT #2
1172      000100      IDE=100     ;INTERRUPT ON DONE ENABLE BIT
1173      000200      RDY=200     ;RP11 CONTROLLER READY BIT
1174      004000      HDR=4000    ;HEADER OPERATION BIT
1175      010000      MODE=10000  ;MODE BIT
1176      020000      AIE=20000  ;ATTENTION INTERRUPT ENABLE BIT
1177      040000      HE=40000   ;HARD ERROR BIT
1178      100000      ERR=100000 ;ERROR BIT
1179
1180
1181

```

COMMON TAGS

.SBTTL COMMON TAGS

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

1182								
1183								
1184								
1185								
1186								
1187								
1188		001100						
1189	001100	000000	\$CMTAG:	0			START OF COMMON TAGS	
1190	001100	000000	\$PASS:	0			CONTAINS PASS COUNT	
1191	001102	000	\$STNM:	0			CONTAINS THE TEST NUMBER	
1192	001103	000	\$ERFLG:	0			CONTAINS ERROR FLAG	
1193	001104	000000	\$ICNT:	0			CONTAINS SUBTEST ITERATION COUNT	
1194	001106	000000	\$LPADR:	0			CONTAINS SCOPE LOOP ADDRESS	
1195	001110	000000	\$LPERR:	0			CONTAINS SCOPE RETURN FOR ERRORS	
1196	001112	000000	\$ERTTL:	0			CONTAINS TOTAL ERRORS DETECTED	
1197	001114	000	\$ITEMB:	0			CONTAINS ITEM CONTROL BYTE	
1198	001115	001	\$ERMAX:	1			CONTAINS MAX. ERRORS PER TEST	
1199	001116	000000	\$ERRPC:	0			CONTAINS PC OF LAST ERROR INSTRUCTION	
1200	001120	000000	\$GDADR:	0			CONTAINS ADDRESS OF 'GOOD' DATA	
1201	001122	000000	\$BDADR:	0			CONTAINS ADDRESS OF 'BAD' DATA	
1202	001124	000000	\$GDDAT:	0			CONTAINS 'GOOD' DATA	
1203	001126	000000	\$BDDAT:	0			CONTAINS 'BAD' DATA	
1204	001130	000000		0			RESERVED--NOT TO BE USED	
1205	001132	000000		0				
1206	001134	000	\$AUTOB:	0			AUTOMATIC MODE INDICATOR	
1207	001135	000	\$INTAG:	0			INTERRUPT MODE INDICATOR	
1208	001136	000000		0				
1209	001140	177570	\$SWR:	0	DSWR		ADDRESS OF SWITCH REGISTER	
1210	001142	177570	\$DISPLAY:	0	DDISP		ADDRESS OF DISPLAY REGISTER	
1211	001144	177560	\$TKS:	0			TTY KBD STATUS	
1212	001146	177562	\$TKB:	0			TTY KBD BUFFER	
1213	001150	177564	\$TPS:	0			TTY PRINTER STATUS REG. ADDRESS	
1214	001152	177566	\$TPB:	0			TTY PRINTER BUFFER REG. ADDRESS	
1215	001154	000	\$NULL:	0			CONTAINS NULL CHARACTER FOR FILLS	
1216	001155	002	\$FILLS:	2			CONTAINS # OF FILLER CHARACTERS REQUIRED	
1217	001156	012	\$FILLC:	12			INSERT FILL CHARS. AFTER A "LINE FEED"	
1218	001157	000	\$TPFLG:	0			"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)	
1219	001160	000000	\$TMPO:	0			USER DEFINED	
1220	001162	000000	\$TIMES:	0			MAX. NUMBER OF ITERATIONS	
1221	001164	000000	\$ESCAPE:	0			ESCAPE ON ERROR ADDRESS	
1222	001166	177607	\$BELL:	0	<207><377><377>		CODE FOR BELL	
1223	001172	077	\$QUES:	0	/?/		QUESTION MARK	
1224	001173	015	\$CRLF:	0	<15>		CARRIAGE RETURN	
1225	001174	000012	\$LF:	0	<12>		LINE FEED	
1226			*****				*****	
1227	001176	000000	\$CHKDRV:	0			ADDRESS OF DRIVE BEING USED	
1228	001200	000000	\$BYPASS:	0			TEST EXIT ADDRESS	
1229	001202	000000	\$FLAG:	0			INTERNAL PROGRAM FLAG WORD	
1230	001204	000000	\$PATNUM:	0			CURRENTLY SELECTED PATTERN	
1231	001206	000000	\$CNTRLC:	0			CONVERSATION MODE FLAG	
1232	001210	000000	\$MAXCYL:	0			MAXIMUM CYLINDER ADDRESS FOR THE DRIVE IN USE	
1233	001212	001750	\$TIMOUT:	1000.			TIME ALLOWED FOR AN OPERATION	
1234	001214	000000	\$ICNT:	0			TEST ITERATION COUNT	
1235	001216	000000	\$BUSADR:	0			CHANGE RP11 ADDRESS FLAG	
1236	001220	000000	\$STALL:	0			STALL VALUE	
1237	001222	000000	\$MMACTV:	0			MEMORY MANAGEMENT IS ACTIVE FLAG	

000377

DZRPYC.P11 09-SEP-77 16:03

COMMON TAGS

1238	001224	000000	000000	ACTMEM: .WORD	0,0	; PHYSICAL ADDRESS OF NEXT READ BUFFER
1239	001230	000000	000000	MAXMEM: .WORD	0,0	; ADDRESS OF LAST MEMORY BANK (FILLED BY \$SIZE)
1240	001234	000000	000000	HIMEM: .WORD	0,0	; PHYSICAL ADDRESS OF LAST MEMORY BANK (FOR TEST 14)
1241	001240	000000	000000	LOMEM: .WORD	0,0	; PHYSICAL ADDRESS OF FIRST BUFFER LOCATION
1242	001244	000000	000000	PAKSIZ: .WORD	0,0	; SECTORS ON THE PACK ON THE DRIVE BEING TESTED
1243	001250	000000		INCSEC: .WORD	0	; SECTOR INCREMENT VALUE
1244	001252	000000		INCTRK: .WORD	0	; TRACK INCREMENT VALUE
1245	001254	000000		BLKS14: .WORD	0	; NUMBER SECTORS IN TRANSFER IN TEST 14
1246	001256	000000		WC14: .WORD	0	; WORD COUNT FOR TEST 14
1247	001260	000000		OPRSEL: .WORD	0	; OPERATOR SELECTED ADDRESS FLAG
1248	001262	020000		LMTBK: .WORD	8192.	; LARGEST TRANSFER PERMITTED
1249	001264	000000		MAXWC: .WORD	0	; MAXIMUM WORD COUNT ALLOWED ON THIS SYSTEM
1250	001266	000000		SCYL: .WORD	0	; OPERATOR SELECTED CYLINDER
1251	001270	000000		STRK: .WORD	0	; OPERATOR SELECTED TRACK
1252	001272	000000		SSEC: .WORD	0	; OPERATOR SELECTED SECTOR
1253	001274	000000		RDERR: .WORD	0	; READ RETRY COUNTER
1254	001276	000000		TSTNMS: .WORD	0	; TEST SELECTION WORD. BIT 0 = TEST 0, ETC.
1255	001300	000000		PATRN: .WORD	0	; PATTERN SELECTION WORD FOR TEST 14
1256	001302	000000		MEMSIZ: .WORD	0	; ADDRESS OF HIGHEST NON-MEMORY MANAGEMENT
1257						; MEMORY LOCATION
1258	001304	000000		DRVTYP: .WORD	0	; CONTAINS A BIT FOR EACH RPO3 DRIVE
1259						; ON THE SYSTEM. BIT00 = DRIVE 0, ETC.
1260	001306	000000		DRVMSK: .WORD	0	; DRIVE TEST SELECTION MASK
1261	001310	000000		DRVBAD: .WORD	0	; CONTAINS A BIT FOR THE DRIVE IF THE DRIVE
1262						; GOES OFFLINE OR BECOMES UNSAFE DURING TESTING.
1263						; BIT00 = DRIVE 0, ETC.
1264	001312	000000		DRVSEL: .WORD	0	; CONTAINS A BIT FOR EACH DRIVE TO BE TESTED
1265	001314	000000		RETRY: .WORD	0	; RETRY COUNTER FOR TEST 14
1266	001316	000003		LARETRY: .WORD	3	; RETRY LIMIT FOR TEST 14
1267	001320	000003		CMP CT: .WORD	3	; NUMBER OF COMPARE ERROR TYPEOUTS
1268	001322	000000		LDRFLG: .WORD	0	; WHEN =0 LOADERS ARE AT TOP OF MEMORY

;RP11 ADDRESS VALUES

1270						
1271						
1272	001324	176710		RPADR: .WORD	176710	; RP11 BUS ADDRESS
1273	001326	000254	000256	RPVEC: .WORD	257,256	; RP11 VECTOR ADDRESS
1274	001332	000240		RPPRIO: .WORD	(5*32.)	; RP11 PRIORITY

;SAVE THE RP11E REGISTERS HERE

1275						
1276						
1277						
1278	001334	000000		\$RPDS: .WORD	0	; DRIVE STATUS REGISTER
1279	001336	000000		\$RPER: .WORD	0	; ERROR REGISTER
1280	001340	000000		\$RPCS: .WORD	0	; COMMAND & STATUS REGISTER
1281	001342	000000		\$RPWC: .WORD	0	; WORD COUNT REGISTER
1282	001344	000000		\$RPBA: .WORD	0	; BUFFER ADDRESS REGISTER
1283	001346	000000		\$RPCA: .WORD	0	; CURRENT CYLINDER ADDRESS REGISTER
1284	001350	000000		\$RPDA: .WORD	0	; TRACK-SECTOR ADDRESS REGISTER
1285	001352	000000		\$RPM1: .WORD	0	; MAINTENANCE REGISTER #1
1286	001354	000000		\$SUCA: .WORD	0	; SELECTED UNIT CYLINDER ADDRESS REGISTER
1287	001356	000000		\$SILO: .WORD	0	; SILO REGISTER

;ATTENTION BITS

1288						
1289						
1290						
1291	001360	001		ATABIT: .BYTE	1	; DRIVE 0
1292	001361	002		.BYTE	2	; DRIVE 1
1293	001362	004		.BYTE	4	; DRIVE 2

: PATTERNS 0 THRU 15

PAT0: .WORD 000000 ; PATTERN 0

.WORD 000000
.WORD 000000
.WORD 000000
.WORD 000000
.WORD 000000
.WORD 000000
.WORD 000000
.WORD 000000
.WORD 000000
.WORD 000000
.WORD 000000
.WORD 000000
.WORD 000000
.WORD 000000
.WORD 000000
.WORD 000000

PAT1: .WORD 000001 ; PATTERN 1

.WORD 000003
.WORD 000007
.WORD 000017
.WORD 000037
.WORD 000077
.WORD 000177
.WORD 000377
.WORD 000777
.WORD 001777
.WORD 003777
.WORD 007777
.WORD 017777
.WORD 037777
.WORD 077777
.WORD 177777

PAT2: .WORD 177776 ; PATTERN 2

.WORD 177774
.WORD 177770
.WORD 177760
.WORD 177740
.WORD 177700
.WORD 177600
.WORD 177400
.WORD 177000
.WORD 176000
.WORD 174000
.WORD 170000
.WORD 160000
.WORD 140000
.WORD 100000
.WORD 000000

PAT3: .WORD 00000C ; PATTERN 3

.WORD 00000C

1
1369 001540 000001
1370 001543 000003
1371 001544 000007
1372 001545 000017
1373 001546 000037
1374 001547 000077
1375 001548 000177
1376 001550 000377
1377 001551 000777
1378 001552 001777
1379 001554 003777
1380 001556 007777
1381 001570 017777
1382 001572 037777
1383 001574 077777
1384 001576 177777
1385
1386 001600 177776
1387 001602 177774
1388 001604 177770
1389 001606 177760
1390 001610 177740
1391 001612 177700
1392 001614 177600
1393 001616 177400
1394 001620 177000
1395 001622 176000
1396 001624 174000
1397 001626 170000
1398 001630 160000
1399 001632 140000
1400 001634 100000
1401 001636 000000
1402
1403
1404 001640 000000
1405 001642 000000

1406	001644	000000	.WORD	000000
1407	001646	177777	.WORD	177777
1408	001650	177777	.WORD	177777
1409	001652	177777	.WORD	177777
1410	001654	000000	.WORD	000000
1411	001656	000000	.WORD	000000
1412	001658	177777	.WORD	177777
1413	001660	177777	.WORD	177777
1414	001662	177777	.WORD	177777
1415	001664	000000	.WORD	000000
1416	001666	177777	.WORD	177777
1417	001670	000000	.WORD	000000
1418	001672	177777	.WORD	177777
1419	001674	177777	.WORD	177777
1420	001676	000000	.WORD	000000
1421	001700	000000	.WORD	000000
1422	001702	010421	.WORD	010421
1423	001704	021042	.WORD	021042
1424	001706	031463	.WORD	031463
1425	001710	042104	.WORD	042104
1426	001712	052525	.WORD	052525
1427	001714	063146	.WORD	063146
1428	001716	073567	.WORD	073567
1429	001720	104210	.WORD	104210
1430	001722	114631	.WORD	114631
1431	001724	125252	.WORD	125252
1432	001726	135673	.WORD	135673
1433	001728	146314	.WORD	146314
1434	001730	156735	.WORD	156735
1435	001732	167356	.WORD	167356
1436	001734	177777	.WORD	177777
1437	001736	177777	.WORD	177777
1438	001740	052525	.WORD	052525
1439	001742	063146	.WORD	063146
1440	001744	073567	.WORD	073567
1441	001746	084188	.WORD	084188
1442	001748	094809	.WORD	094809
1443	001750	105430	.WORD	105430
1444	001752	116051	.WORD	116051
1445	001754	126672	.WORD	126672
1446	001756	137293	.WORD	137293
1447	001760	147914	.WORD	147914
1448	001762	158535	.WORD	158535
1449	001764	169156	.WORD	169156
1450	001766	179777	.WORD	179777
1451	001770	125252	.WORD	125252
1452	001772	135673	.WORD	135673
1453	001774	146314	.WORD	146314
1454	001776	156735	.WORD	156735
1455	002000	007417	.WORD	007417
1456	002002	007417	.WORD	007417
1457	002004	007417	.WORD	007417
1458	002006	170360	.WORD	170360
1459	002010	170360	.WORD	170360
1460	002012	170360	.WORD	170360
1461	002014	007417	.WORD	007417

PAT4: .WORD 000000 ;PATTERN 4

PAT5: .WORD 052525 ;PATTERN 5

PAT6: .WORD 007417 ;PATTERN 6

1460	002016	007417	.WORD	007417
1461	002016	170360	.WORD	170360
1462	002016	170360	.WORD	170360
1463	002016	170360	.WORD	170360
1464	002016	007417	.WORD	007417
1465	002016	170360	.WORD	170360
1466	002016	007417	.WORD	007417
1467	002016	170360	.WORD	170360
1468	002016	170360	.WORD	170360
1469	002016	170360	.WORD	170360
1470	002016	007417	.WORD	007417
1471	002040	026455	.WORD	026455
1472	002040	026455	.WORD	026455
1473	002040	026455	.WORD	026455
1474	002040	026455	.WORD	026455
1475	002046	151322	.WORD	151322
1476	002046	151322	.WORD	151322
1477	002046	151322	.WORD	151322
1478	002046	151322	.WORD	151322
1479	002050	151322	.WORD	151322
1480	002050	151322	.WORD	151322
1481	002050	151322	.WORD	151322
1482	002050	151322	.WORD	151322
1483	002054	026455	.WORD	026455
1484	002054	026455	.WORD	026455
1485	002054	026455	.WORD	026455
1486	002054	026455	.WORD	026455
1487	002056	151322	.WORD	151322
1488	002056	151322	.WORD	151322
1489	002056	151322	.WORD	151322
1490	002056	151322	.WORD	151322
1491	002066	026455	.WORD	026455
1492	002066	151322	.WORD	151322
1493	002066	151322	.WORD	151322
1494	002070	026455	.WORD	026455
1495	002070	151322	.WORD	151322
1496	002070	151322	.WORD	151322
1497	002074	151322	.WORD	151322
1498	002074	151322	.WORD	151322
1499	002076	026455	.WORD	026455
1500	002100	077577	.WORD	077577
1501	002100	077577	.WORD	077577
1502	002102	077577	.WORD	077577
1503	002104	077577	.WORD	077577
1504	002106	077577	.WORD	077577
1505	002110	077577	.WORD	077577
1506	002112	077577	.WORD	077577
1507	002114	077577	.WORD	077577
1508	002116	077577	.WORD	077577
1509	002120	077577	.WORD	077577
1510	002122	077577	.WORD	077577
1511	002124	077577	.WORD	077577
1512	002126	077577	.WORD	077577
1513	002130	077577	.WORD	077577
1514	002132	077577	.WORD	077577
1515	002134	077577	.WORD	077577
1516	002136	077577	.WORD	077577
1517	002140	000001	.WORD	000001
1518	002142	000002	.WORD	000002
1519	002144	000004	.WORD	000004
1520	002146	000010	.WORD	000010
1521	002150	000020	.WORD	000020
1522	002152	000040	.WORD	000040
1523	002154	000100	.WORD	000100
1524	002156	000200	.WORD	000200
1525	002160	000400	.WORD	000400
1526	002162	001000	.WORD	001000
1527	002164	002000	.WORD	002000
1528	002166	004000	.WORD	004000

PAT7: .WORD 026455 ;PATTERN 7

PAT8: .WORD 077577 ;PATTERN 8 (WORST CASE PATTERN)

PAT9: .WORD 000001 ;PATTERN 9

1518	002170	010000	.WORD	010000
1519	002172	020000	.WORD	020000
1520	002174	040000	.WORD	040000
1521	002176	100000	.WORD	100000
1522	002178	177776	.WORD	177776
1523	002180	177775	.WORD	177775
1524	002182	177773	.WORD	177773
1525	002184	177767	.WORD	177767
1526	002186	177757	.WORD	177757
1527	002188	177737	.WORD	177737
1528	002190	177677	.WORD	177677
1529	002192	177577	.WORD	177577
1530	002194	177377	.WORD	177377
1531	002196	176777	.WORD	176777
1532	002198	175777	.WORD	175777
1533	002200	173777	.WORD	173777
1534	002202	167777	.WORD	167777
1535	002204	157777	.WORD	157777
1536	002206	137777	.WORD	137777
1537	002208	077777	.WORD	077777
1538	002210	172666	.WORD	172666
1539	002212	155555	.WORD	155555
1540	002214	172666	.WORD	172666
1541	002216	155555	.WORD	155555
1542	002218	172666	.WORD	172666
1543	002220	155555	.WORD	155555
1544	002222	172666	.WORD	172666
1545	002224	155555	.WORD	155555
1546	002226	172666	.WORD	172666
1547	002228	155555	.WORD	155555
1548	002230	172666	.WORD	172666
1549	002232	155555	.WORD	155555
1550	002234	172666	.WORD	172666
1551	002236	155555	.WORD	155555
1552	002238	172666	.WORD	172666
1553	002240	155555	.WORD	155555
1554	002242	172666	.WORD	172666
1555	002244	155555	.WORD	155555
1556	002300	077777	.WORD	077777
1557	002302	137777	.WORD	137777
1558	002304	157777	.WORD	157777
1559	002306	167777	.WORD	167777
1560	002310	173777	.WORD	173777
1561	002312	175777	.WORD	175777
1562	002314	176777	.WORD	176777
1563	002316	177377	.WORD	177377
1564	002320	177577	.WORD	177577
1565	002322	177677	.WORD	177677
1566	002324	177737	.WORD	177737
1567	002326	177757	.WORD	177757
1568	002330	177767	.WORD	177767
1569	002332	177773	.WORD	177773
1570	002334	177775	.WORD	177775
1571	002336	177776	.WORD	177776

PAT10: .WORD 177776 ;PATTERN 10

PAT11: .WORD 172666 ;PATTERN 11

PAT12: .WORD 077777 ;PATTERN 12

1574	002340	177777	PAT13:	.WORD	177777	:PATTERN 13
1575	002342	177777		.WORD	177777	
1576	002344	177777		.WORD	177777	
1577	002346	177777		.WORD	177777	
1578	002348	177777		.WORD	177777	
1579	002350	177777		.WORD	177777	
1580	002352	177777		.WORD	177777	
1581	002354	177777		.WORD	177777	
1582	002356	177777		.WORD	177777	
1583	002358	177777		.WORD	177777	
1584	002360	177777		.WORD	177777	
1585	002362	177777		.WORD	177777	
1586	002364	177777		.WORD	177777	
1587	002366	177777		.WORD	177777	
1588	002368	177777		.WORD	177777	
1589	002370	177777		.WORD	177777	
1590	002372	177777		.WORD	177777	
1591	002374	177777		.WORD	177777	
1592	002400	000000	PAT 4:	.WORD	000000	:PATTERN 14
1593	002402	000000		.WORD	000000	
1594	002404	177777		.WORD	177777	
1595	002406	177777		.WORD	177777	
1596	002408	177777		.WORD	177777	
1597	002410	177777		.WORD	177777	
1598	002412	177777		.WORD	177777	
1599	002414	177777		.WORD	177777	
1599	002416	177777		.WORD	177777	
1600	002418	177777		.WORD	177777	
1600	002420	177777		.WORD	177777	
1601	002422	177777		.WORD	177777	
1602	002424	177777		.WORD	177777	
1603	002426	177777		.WORD	177777	
1604	002428	177777		.WORD	177777	
1605	002430	177777		.WORD	177777	
1606	002432	177777		.WORD	177777	
1607	002434	177777		.WORD	177777	
1608	002436	177777		.WORD	177777	
1609	002438	177777		.WORD	177777	
1610	002440	000000	PAT15:	.WORD	177777	:PATTERN 15
1611	002442	000000		.WORD	177777	
1612	002444	000000		.WORD	000000	
1613	002446	000000		.WORD	000000	
1614	002448	000000		.WORD	000000	
1615	002450	000000		.WORD	000000	
1616	002452	000000		.WORD	000000	
1617	002454	000000		.WORD	000000	
1618	002456	000000		.WORD	000000	
1619	002458	000000		.WORD	000000	
1620	002460	000000		.WORD	000000	
1621	002462	000000		.WORD	000000	
1622	002464	000000		.WORD	000000	
1623	002466	000000		.WORD	000000	
1624				.WORD	000000	
1625			:DRIVE OPERATION CONTROL BLOCK			
1626						
1627	002500	000	CPB:	.BYTE	C	:DRIVE NUMBER
1628	002501	000		.BYTE	C	: 'MEX' BITS
1629	002502	000		.BYTE	C	: OPERATION CODE

1630	002503	000
1631	002504	000000
1632	002506	000000
1633	002510	000000
1634	002512	000
1635	002513	000
1636		

SWC:	.BYTE	0
SBUF:	.WORD	000000
SCYL:	.WORD	000000
SSEC:	.BYTE	000
STRK:	.BYTE	000

:	MODE AND HOR BITS
:	WORD COUNT
:	BUFFER ADDRESS
:	CYLINDER ADDRESS
:	SECTOR ADDRESS
:	TRACK ADDRESS

ERROR POINTER TABLE

.SBTTL ERROR POINTER TABLE

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

:* EM ::POINTS TO THE ERROR MESSAGE
:* OH ::POINTS TO THE DATA HEADER
:* OT ::POINTS TO THE DATA
:* DF ::POINTS TO THE DATA FORMAT

\$ERRTB:
:ERROR 1

:RP11 DIDN'T RESPOND TO ADDRESSING

:ERROR 2

:CAN'T START READ COMMAND

:ERROR 3

: 'CLEAR' COMMAND DIDN'T TERMINATE DATA TRANSFER OPERATION

:ERROR 4

:ERROR WRITING TEST SECTOR(S)

:ERROR 5

:WRITE CHECK ERROR CHECKING TEST SECTOR(S)

:ERROR 6

:EXPECTED 'WCE' DIDN'T OCCUR

1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651 002514
1652
1653
1654 002514 032173
1655 002516 040315
1656 002520 04303E
1657 002522 043516
1658
1659
1660
1661 002524 032235
1662 002526 040323
1663 002530 043040
1664 002532 043522
1665
1666
1667
1668 002534 032266
1669 002536 040323
1670 002540 043040
1671 002542 043522
1672
1673
1674
1675 002544 032357
1676 002546 040323
1677 002550 043040
1678 002552 043522
1679
1680
1681
1682 002554 032414
1683 002556 040465
1684 002560 043072
1685 002562 043532
1686
1687
1688
1689 002564 032466
1690 002566 040465
1691 002570 043072
1692 002572 043532

EM1
OH1
OT1
DF1

EM2
OH2
OT2
DF2

EM3
OH3
OT3
DF3

EM4
OH4
OT4
DF4

EM5
OH5
OT5
DF5

EM6
OH6
OT6
DF6

1693				
1694			:ERROR 7	
1695				
1696	002574	032522	EM7	:ERROR WRITING LESS THAN A FULL SECTOR
1697	002576	040323	DH2	
1698	002600	043040	DT2	
1699	002602	043522	DF2	
1700				
1701			:ERROR 10	
1702				
1703	002604	000000	0	:UNUSED
1704	002606	000000	0	
1705	002610	000000	0	
1706	002612	000000	0	
1707				
1708			:ERROR 11	
1709				
1710	002614	032570	EM11	:ONE OF 2 WORDS WRITTEN IN THE PARTIAL WRITE DIDN'T COMPARE
1711	002616	040637	DH11	
1712	002620	043126	DT11	
1713	002622	043542	DF11	
1714				
1715			:ERROR 12	
1716				
1717	002624	032663	EM12	:NON-ZERO DATA IN PART OF SECTOR WHICH SHOULD HAVE :BEEN ZERO FILLED DURING PARTIAL WRITE
1718				
1719	002626	040720	DH12	
1720	002630	043142	DT12	
1721	002632	043542	DF11	
1722				
1723			:ERROR 13	
1724				
1725	002634	033014	EM13	: 'EOP' DIDN'T SET DURING A 2 SECTOR WRITE BEGINNING :AT THE LAST SECTOR OF THE PACK
1726				
1727	002636	040323	DH2	
1728	002640	043040	DT2	
1729	002642	043522	DF2	
1730				
1731			:ERROR 14	
1732				
1733	002644	033137	EM14	: 'ERR' DIDN'T SET WITH 'EOP'
1734	002646	040775	DH14	
1735	002650	043156	DT14	
1736	002652	043542	DF11	
1737				
1738			:ERROR 15	
1739				
1740	002654	033173	EM15	: 'EOP' DIDN'T CLEAR RPOA
1741	002656	040323	DH2	
1742	002660	043040	DT2	
1743	002662	043522	DF2	
1744				
1745			:ERROR 16	
1746				
1747	002664	033223	EM16	: SUCA NOT CORRECT AFTER 'EOP'
1748	002666	041052	DH16	

ERROR POINTER TABLE

1749	002670	043172	DT16	
1750	002672	043616	DF53	
1751				: ERROR 17
1752				
1753				
1754	002674	033260	EM17	: ERROR READING TEST SECTOR
1755	002676	040323	DH2	
1756	002700	043040	DT2	
1757	002702	043522	DF2	
1758				
1759				: ERROR 20
1760				
1761	002704	033312	EM20	: CONTENTS OF FIRST SECTOR OF PACK CHANGED AFTER
1762				: FORCING 'EOP' DURING WRITE
1763	002706	040637	DH11	
1764	002710	043126	DT11	
1765	002712	043542	DF11	
1766				
1767				: ERROR 21
1768				
1769	002714	033425	EM21	: 'PROG' ERROR DIDN'T SET WHEN READ COMMAND ISSUED
1770				: WHILE CONTROLLER BUSY
1771	002716	040775	DH14	
1772	002720	043156	DT14	
1773	002722	043542	DF11	
1774				
1775				: ERROR 22
1776				
1777	002724	033535	EM22	: 'ERR' DIDN'T SET WITH 'PROG'
1778	002726	040775	DH14	
1779	002730	043156	DT14	
1780	002732	043542	DF11	
1781				
1782				: ERROR 23
1783				
1784	002734	033572	EM23	: ERROR ATTEMPTING TO MISFORMAT SECTOR 0
1785	002736	040323	DH2	
1786	002740	043040	DT2	
1787	002742	043522	DF2	
1788				
1789				: ERROR 24
1790				
1791	002744	033641	EM24	: ERROR VERIFYING THE MISFORMATTED TEST HEADER
1792	002746	040775	DH14	
1793	002750	043156	DT14	
1794	002752	043542	DF11	
1795				
1796				: ERROR 25
1797				
1798	002754	033716	EM25	: MISFORMATTED TEST HEADER IS NOT CORRECT
1799	002756	041167	DH25	
1800	002760	043204	DT25	
1801	002762	043542	DF25	
1802				
1803				: ERROR 26
1804				

ERROR POINTER TABLE

1805	002764	033766	EM26	: 'HNF' DIDN'T SET WHEN SEARCHING FOR MISFORMATTED SECTOR
1806	002766	040323	DH2	
1807	002770	043040	DT2	
1808	002772	043522	DF2	
1809				
1810				: ERROR 27
1811				
1812	002774	034056	EM27	: 'DSK' NOT SET WITH 'HNF'
1813	002776	040775	DH14	
1814	003000	043156	DT14	
1815	003002	043542	DF11	
1816				
1817				: ERROR 30
1818				
1819	003004	034107	EM30	: 'HE' DIDN'T SET WITH 'HNF'
1820	003006	040775	DH14	
1821	003010	043156	DT14	
1822	003012	043542	DF11	
1823				
1824				: ERROR 31
1825				
1826	003014	034142	EM31	: 'ERR' NOT SET WITH 'HNF'
1827	003016	040775	DH14	
1828	003020	043156	DT14	
1829	003022	043542	DF11	
1830				
1831				: ERROR 32
1832				
1833	003024	034173	EM32	: ERROR WHILE RESTORING MISFORMATTED TEST HEADER
1834	003026	040323	DH2	
1835	003030	043040	DT2	
1836	003032	043522	DF2	
1837				
1838				: ERROR 33
1839				
1840	003034	034252	EM33	: CONTROLLER DIDN'T BECOME BUSY WHEN READ COMMAND : ISSUED TO SEEKING DRIVE
1841				
1842	003036	040323	DH2	
1843	003040	043040	DT2	
1844	003042	043522	DF2	
1845				
1846				: ERROR 34
1847				
1848	003044	034363	EM34	: ERROR DURING READ COMMAND WHICH WAS ISSUED TO SEEKING DRIVE
1849	003046	040323	DH2	
1850	003050	043040	DT2	
1851	003052	043522	DF2	
1852				
1853				: ERROR 35
1854				
1855	003054	034457	EM35	: DATA INCORRECT FROM HEADER READ COMMAND ISSUED : TO SEEKING DRIVE
1856				
1857	003056	041167	DH25	
1858	003060	043204	DT25	
1859	003062	043546	DF25	
1860				

1861				:ERROR 36	
1862					
1863	003064	034561	EM36		;'NXME' DIDN'T SET WHEN LOCATION 760000 ADDRESSED
1864	003066	040323	DH2		
1865	003070	043040	DT2		
1866	003072	043522	DF2		
1867					
1868				:ERROR 37	
1869					
1870	003074	034642	EM37		;'HE' DIDN'T SET WITH 'NXME'
1871	003076	040775	DH14		
1872	003100	043156	DT14		
1873	003102	043542	DF11		
1874					
1875				:ERROR 40	
1876					
1877	003104	034676	EM40		;'ERR' DIDN'T SET WITH 'NXME'
1878	003106	040775	DH14		
1879	003110	043156	DT14		
1880	003112	043542	DF11		
1881					
1882				:ERROR 41	
1883					
1884	003114	034733	EM41		;ERROR FORMATTING TRACK 0
1885	003116	040323	DH2		
1886	003120	043040	DT2		
1887	003122	043522	DF2		
1888					
1889				:ERROR 42	
1890					
1891	003124	034764	EM42		;ERROR READING THE HEADER FROM THE TEST SECTOR
1892	003126	040323	DH2		
1893	003130	043040	DT2		
1894	003132	043522	DF2		
1895					
1896				:ERROR 43	
1897					
1898	003134	035042	EM43		;SECTOR FIELD FROM HEADER IS NOT CORRECT
1899	003136	041256	DH43		
1900	003140	043224	DT43		
1901	003142	043556	DF43		
1902					
1903				:ERROR 44	
1904					
1905	003144	035112	EM44		;ERROR WRITING SECTOR ADDRESS IN DATA FIELD
1906	003146	040323	DH2		
1907	003150	043040	DT2		
1908	003152	043522	DF2		
1909					
1910				:ERROR 45	
1911					
1912	003154	035165	EM45		;'SOT' OR SECTOR ADDRESS REGISTER IS NOT CORRECT
1913	003156	041502	DH45		
1914	003160	043262	DT45		
1915	003162	043572	DF45		
1916					

1917				; ERROR 46	
1918					
1919	003164	035245	EM46		; DATA NOT CORRECT FOR SECTOR READ
1920	003166	041674	DM46		
1921	003170	043306	DT46		
1922	003172	043542	DF11		
1923					
1924				; ERROR 47	
1925					
1926	003174	035316	EM47		; SECTOR CONTENTS WRONG, DATA SHOULD BE THE
1927					; TRACK NUMBER OF THE CURRENTLY SELECTED HEAD
1928	003176	042031	DM47		
1929	003200	043322	DT47		
1930	003202	043602	DF47		
1931					
1932				; ERROR 50	
1933					
1934	003204	035445	EM50		; ERROR AFTER 2 WORD READ STARTING AT LOC 177776
1935	003206	040323	DM2		
1936	003210	043040	DT2		
1937	003212	043522	DF2		
1938					
1939				; ERROR 51	
1940					
1941	003214	035524	EM51		; 'MEX0' DIDN'T SET AFTER 2 WORD READ STARTING
1942					; AT LOC 177776
1943	003216	042145	DM51		
1944	003220	043346	DT51		
1945	003222	043612	DF51		
1946					
1947				; ERROR 52	
1948					
1949	003224	035620	EM52		; 'MEX1' SET AFTER 2 WORD READ STARTING AT LOC 177776
1950	003226	042145	DM51		
1951	003230	043346	DT51		
1952	003232	043612	DF51		
1953					
1954				; ERROR 53	
1955					
1956	003234	035704	EM53		; CONTENTS OF LOC 200000 WRONG AFTER 2 WORD READ
1957					; STARTING AT LOC 177776
1958	003236	041167	DM25		
1959	003240	043204	DT25		
1960	003242	043616	DF53		
1961					
1962				; ERROR 54	
1963					
1964	003244	036013	EM54		; ERROR AFTER 2 WORD READ STARTING AT LOC 377776
1965	003246	040323	DM2		
1966	003250	043040	DT2		
1967	003252	043522	DF2		
1968					
1969				; ERROR 55	
1970					
1971					
1972	003254	036072	EM55		; 'MEX0' DIDN'T CLEAR AFTER 2 WORD STARTING AT LOC 377776

ERROR POINTER TABLE

1973	003256	0432145	DM51	
1974	003258	043346	DT51	
1975	003262	043612	DF51	
			: ERROR 56	
1976	003264	036162	EM56	: 'MEX1' DIDN'T SET AFTER 2 WORD REAC STARTING AT LOC 377776
1977	003266	043346	DM51	
1978	003268	043346	DT51	
1979	003272	043612	DF51	
			: ERROR 57	
1980	003274	036255	EM57	: CONTENTS OF LOC 400000 WRONG AFTER 2 WORD REAC
1981	003276	041167	DM25	: STARTING AT LOCATION 377776
1982	003300	043204	DT25	
1983	003302	043616	DF53	
			: ERROR 60	
1984	003304	036371	EM60	: ERROR RETURNING SECTOR TO PDP-11 MODE USING A
1985	003306	040323	DM2	: 2 WORD WRITE
1986	003310	043040	DT2	
1987	003312	043522	DF2	
			: ERROR 61	
1988	003314	036465	EM61	: ERROR WRITING TEST SECTOR(S)
1989	003316	040465	DM61	
1990	003320	043072	DT61	
1991	003322	043322	DF61	
			: ERROR 62	
1992	003324	036522	EM62	: ERROR READING TEST SECTOR(S)
1993	003326	040465	DM62	
1994	003330	042072	DT62	
1995	003332	043322	DF62	
			: ERROR 63	
1996	003334	036557	EM63	: DRIVE DIDN'T RETURN TO CYL 0 FROM THE MAXIMUM
1997	003336	042242	DM63	: CYL ON CONTROLLER POWER FAIL
1998	003340	043366	DT63	
1999	003342	043622	DF63	
			: ERROR 64	
2000	003344	036673	EM64	: CONTENTS OF MEMORY CHANGED DURING POWER
2001	003346	040720	DM12	: FAIL WHILE READING THE DISK
2002	003350	043142	DT12	
2003	003352	043542	DF11	

ERROR POINTER TABLE

2060			:ERROR 65		
2061	003354	037000		EM65	: 'SUOL' SET WITH DRIVE DISABLED
2062	003356	040775		OH14	
2063	003360	043156		DT14	
2064	003362	043542		DF11	
2065			:ERROR 66		
2066	003364	037037		EM66	: 'SUOL' NOT SET WITH DRIVE ENABLED
2067	003366	040775		OH14	
2068	003370	043156		DT14	
2069	003372	043542		DF11	
2070			:ERROR 67		
2071	003374	037101		EM67	: 'SUWP' NOT SET WITH DRIVE SET TO 'READ ONLY'
2072	003376	040775		OH14	
2073	003400	043156		DT14	
2074	003402	043542		DF11	
2075			:ERROR 70		
2076	003404	037156		EM70	: 'SUWP' SET WITH DRIVE SET TO 'READ/WRITE'
2077	003406	040775		OH14	
2078	003410	043156		DT14	
2079	003412	043542		DF11	
2080			:ERROR 71		
2081	003414	037230		EM71	: 'SUWP' SET WITH RP11 'WRITE LOCKOUT' SET. CYLINDER
2082					: LOA'S = 0. AND RPCA = 0
2083	003416	042327		OH71	
2084	003420	043404		DT71	
2085	003422	043622		DF63	
2086			:ERROR 72		
2087	003424	037344		EM72	: 'SUWP' SET WITH RPCA = 2 & CYL LOA'S = 0
2088	003426	042327		OH71	
2089	003430	043404		DT71	
2090	003432	043622		DF63	
2091			:ERROR 73		
2092	003434	037415		EM73	: 'SUWP' NOT SET; RPCA = VALUE IN CYL LOA'S
2093	003436	042327		OH71	
2094	003440	043404		DT71	
2095	003442	043622		DF63	
2096			:ERROR 74		
2097	003444	037467		EM74	: 'SUWP' SET WITH RPCA ONE GREATER THAN CYL LOA VALUE
2098	003446	042327		OH71	
2099	003450	043404		DT71	

ERROR POINTER TABLE

003452	043622	DF63	
		: ERROR 75	
003454	037553	EM75	: 'SUMP' SET WITH DRIVE LOA SWITCHES EQUAL SELECTED DRIVE
003456	040775	DH14	
003460	043156	DT14	
003462	043156	DF11	
		: ERROR 76	
003464	037906	EM76	: DRIVE HAS GONE OFFLINE
003466	040323	DH2	
003470	043040	DT2	
003472	043522	DF2	
		: ERROR 77	
003474	037675	EM77	: DRIVE IS UNSAFE
003476	040323	DH2	
003500	043040	DT2	
003502	043522	DF2	
		: ERROR 100	
003504	037715	EM100	: DRIVE TIMED OUT
003506	040323	DH2	
003510	043040	DT2	
003512	043522	DF2	
		: ERROR 101	
003514	037735	EM101	: CONTROLLER TIMED OUT
003516	040323	DH2	
003520	043040	DT2	
003522	043522	DF2	
		: ERROR 102	
003524	037762	EM102	: DATA COMPARE ERROR
003526	042414	DH102	
003530	043422	DT102	
003532	043626	DF102	
		: ERROR 103	
003534	040005	EM103	: DATA COMPARE ERROR (MEMORY MANAGEMENT) ENABLED
003536	042414	DH102	
003540	043450	DT103	
003542	043636	DF103	
		: ERROR 104	
003544	000000	0	: DATA COMPARE ERROR DETAIL LINE
003546	000000	0	
003550	043504	DT104	

ERROR POINTER TABLE

159	003574	040120	EM107	: 'HE' DIDN'T SET WITH 'PROG'
160	003576	040120	DH14	
161	003578	040120	DT14	
162	003580	040120	DF11	
163				
164				
165				
166	003604	000000	0	: UNSUCCESSFUL AFTER 3 RETRIES
167	003606	042745	DH110	
168	003610	000000	0	
169	003612	000000	0	
170				
171				
172				
173	003614	000000	0	: SUCCESSFUL AFTER 'N' RETRIES
174	003616	043000	DH111	
175	003620	043514	DT111	
176	003622	043516	DF1	
177				
178				
179				
180	003624	040154	EM112	: DRIVE UNSAFE AFTER POWER FAIL
181	003626	040323	DH2	
182	003630	043040	DT2	
183	003632	043522	DF2	
184				
185				
186				
187	003634	040212	EM113	: 'SUMP' NOT SET WITH DRIVE LOA SWITCHES GREATER THAN SELECTED DP
188	003636	040715	DH14	
189	003640	043156	DT14	
190	003642	043542	DF11	

2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246

```

*****
.SBTTL START OF PROGRAM
*****

2201 003644 012737 177777 001216 START3: MOV # -1, BUSADR ; SET BUSADR FLAG
2202 003652 000402 BR START1A ;
2203 003654 005037 001216 START1: CLR # BUSADR ; CLR BUSADR FLAG
2204 003660 005037 001206 START1A: CLR # CNTRLC ; NO CONVERSATION MODE
2205 003664 000411 BR START ;
2206 003666 012737 177777 001216 START4: MOV # -1, BUSADR ; SET BUSADR FLAG
2207 003674 000402 BR START2A ;
2208 003676 005037 001216 START2: CLR # BUSADR ; CLR BUSADR FLAG
2209 003702 012737 177777 001206 START2A: MOV # -1, CNTRLC ; SET CONVERSATION MODE FLAG
2210 003710 000005 START: RESET ; CLEAR THE BUS
2211 .SBTTL INITIALIZE THE COMMON TAGS
2212 ;; CLEAR THE COMMON TAGS ($CMTAG) AREA
2213 003712 012706 001100 MOV # $CMTAG, R6 ; FIRST LOCATION TO BE CLEARED
2214 003716 005026 CLR (R6)+ ; CLEAR MEMORY LOCATION
2215 003720 022706 001140 CMP # SWR, R6 ; ; DONE?
2216 003724 001374 BNE -6 ; ; LOOP BACK IF NO
2217 003726 012706 001100 MOV # STACK, SP ; ; SETUP THE STACK POINTER
2218 ;; INITIALIZE A FEW VECTORS
2219 003732 012737 021646 000020 MOV # $SCOPE, @ IOTVEC ; ; IOT VECTOR FOR SCOPE ROUTINE
2220 003740 012737 000340 000022 MOV # 340, @ IOTVEC+2 ; ; LEVEL 7
2221 003746 012737 017070 000030 MOV # $ERROR, @ EMTVEC ; ; EMT VECTOR FOR ERROR ROUTINE
2222 003754 012737 000340 000032 MOV # 340, @ EMTVEC+2 ; ; LEVEL 7
2223 003762 012737 022502 000034 MOV # $TRAP, @ TRAPVEC ; ; TRAP VECTOR FOR TRAP CALLS
2224 003770 012737 000340 000036 MOV # 340, @ TRAPVEC+2 ; ; LEVEL 7
2225 003776 013737 017004 016776 MOV # $ENDCT, $EOPCT ; ; SETUP END-OF-PROGRAM COUNTER
2226 004004 005037 001162 CLR $TIMES ; ; INITIALIZE NUMBER OF ITERATIONS
2227 004010 005037 001164 CLR $ESCAPE ; ; CLEAR THE ESCAPE ON ERROR ADDRESS
2228 004014 112737 000001 001115 MOVB # 1, $ERMAX ; ; ALLOW ONE ERROR PER TEST
2229 004022 012737 004022 001106 MOV # ., $LPADR ; ; INITIALIZE THE LOOP ADDRESS FOR SCOPE
2230 004030 012737 004030 001110 MOV # ., $LPERR ; ; SETUP THE ERROR LOOP ADDRESS
2231 ;; SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
2232 ;; EQUAL TO A "-1" SETUP FOR A SOFTWARE SWITCH REGISTER.
2233 004036 013746 000004 MOV @ $ERRVEC - (SP) ; ; SAVE ERROR VECTOR
2234 004042 012737 004076 000004 MOV # 64$, @ $ERRVEC ; ; SET UP ERROR VECTOR
2235 004050 012737 177570 001140 MOV # $SWR, SWR ; ; SETUP FOR A HARDWARE SWICH REGISTER
2236 004056 012737 177570 001142 MOV # $DISP, DISPLAY ; ; AND A HARDWARE DISPLAY REGISTER
2237 004064 022777 177777 175046 CMP # -1, @ SWR ; ; TRY TO REFERENCE HARDWARE SWR
2238 004072 001012 BNE 66$ ; ; BRANCH IF NO TIMEOUT TRAP OCCURRED
2239 ; ; AND THE HARDWARE SWR IS NOT = -1
2240 004074 000403 BR 65$ ; ; BRANCH IF NO TIMEOUT
2241 004076 012716 004104 64$: MOV # 65$, (SP) ; ; SET UP FOR TRAP RETURN
2242 004102 000002 RTI ; ;
2243 004104 012737 000176 001140 65$: MOV # $SWREG, SWR ; ; POINT TO SOFTWARE SWR
2244 004112 012737 000174 001142 MOV # $DISPREG, DISPLAY ; ;
2245 004120 012637 000004 66$: MOV (SP)+, @ $ERRVEC ; ; RESTORE ERROR VECTOR
2246 004124 023737 000042 000046 CMP @ 42, @ 46 ; ; ACT11 AUTO MODE

```

2248	004132	001405				BEQ	STARTS	:: YES, SKIP TITLE PRINTOUT
2249	004134	005227	177777			INC	#-1	:: FIRST START ?
2250	004140	001002				BNE	STARTS	:: BR IF NOT
2251	004142	104401	043656			TYPE	TITLE	:: TYPE THE PROGRAM'S MAINDEC NUMBER AND NAME
2252	004146	004737	020416			STARTS:	JSR PC,STKINT	:: SETUP THE TTY KEYBOARD
2253						.SBTTL	GET VALUE	:: SWITCH REGISTER
2254	004152	005737	000042			TST	#042	:: ARE WE RUNNING UNDER XXDP/ACT?
2255	004156	001006				BNE	645	:: BRANCH IF YES
2256	004160	023727	001140	000176		CMP	SWR, #SWREG	:: SOFTWARE SWITCH REG SELECTED?
2257	004166	001005				BNE	655	:: BRANCH IF NO
2258	004170	104406				GTSWR		:: GET SOFT-SWR SETTINGS
2259	004172	000403				BR	655	
2260	004174	112737	000001	001134	645:	MOVB	#1, SAUTOB	:: SET AUTO-MODE INDICATOR
2261	004202				655:			
2262	004202	004737	023452			JSR	PC,RESLDR	:: RESTORE THE LOADER
2263	004206	004737	025700			JSR	PC,SIZMEM	:: SET MEMORY SIZE VARIABLES
2264	004212	005037	001112			CLR	SEATTL	:: CLEAR TOTAL ERROR COUNT
2265	004216	005037	001260			CLR	OPRSEL	:: CLEAR OPERATOR SELECTED ADDRESS FLAG
2266	004222	005037	001266			CLR	SCYL	:: OPERATOR SPECIFIED CYLINDER
2267	004226	005037	001270			CLR	STRK	:: OPERATOR SPECIFIED TRACK
2268	004230	005037	001272			CLR	SSEC	:: OPERATOR SPECIFIED SECTOR
2269	004236	013737	001264	001256		MOV	MAXWC, WC14	:: ASSUME THAT MAXIMUM WORD COUNT LESS THAN 9x
2270	004244	023737	001264	001256		CMP	MAXWC, LMTBK	:: MAXIMUM WORD COUNT GREATER THAN LIMIT
2271	004252	101403				T OS	IS	:: BR IF NOT
2272	004254	013737	001262	001256		MOV	LMTBK, WC14	:: USE AN 8K MAXIMUM
2273	004262	012737	177777	001300	15:	MOV	#177777, PATTRN	:: ENABLE ALL PATTERNS
2274	004270	012737	017777	001276		MOV	#17777, TSTMMS	:: SELECT TESTS 0 - 14
2275	004276	005037	177776			CLR	PSW	:: ENSURE THE PRIORITY = 0
2276	004302	005037	001310			CLR	DRVBAD	:: CLEAR OFFLINE/UNSAFE DRIVE BITS
2277	004306	005037	001162			CLR	STIMES	:: INITIALIZE NUMBER OF ITERATIONS
2278	004312	005037	001164			CLR	SESCAPE	:: CLEAR THE ESCAPE ON ERROR ADDRESS
2279	004316	012737	016616	001200		MOV	#SEOP, BYPASS	:: INITIAL BYPASS ADDRESS
2280	004324	012737	000001	001104		MOV	#1, SICNT	:: PRESET ITERATION COUNT TO 1
2281	004332	112737	000001	001115		MOVB	#1, SERMAX	:: ALLOW ONE ERROR PER TEST
2282	004340	012737	004340	001106		MOV	#, \$LPADR	:: INITIALIZE THE LOOP ADDRESS FOR SCOPE
2283	004346	012737	004346	001110		MOV	#, \$LPERR	:: SETUP THE ERROR LOOP ADDRESS
2284	004354	005037	001222			CLR	MMACTV	:: CLEAR MEMORY MANAGEMENT FLAG
2285	004360	005037	177776			START6:	CLR PS	:: CLEAR THE PROCESSOR PRIORITY
2286	004364	004737	026150			JSR	PC,GETADR	:: CHECK THE RP11 ADDRESS
2287	004370	004737	022566			JSR	PC,RPINIT	:: FIND OUT WHICH DRIVES ARE ON SYSTEM
2288	004374	012737	000312	001210		MOV	#R02, MAXCYL	:: ASSUME RPO2'S
2289	004402	005737	001304			TST	DRVTYP	:: WHICH DRIVES ?
2290	004406	001403				BEQ	25	:: BR IF THEY REALLY WERE RPO2'S
2291	004410	012737	000625	001210		MOV	#405, .MAXCYL	:: SET MAX CYLINDER FOR RPO3'S
2292	004416	005227	177777		25:	INC	#-1	:: FIRST START ?
2293	004422	001404				BEQ	95	:: BR IF IT IS
2294	004424	032777	000200	174506		BIT	#SW07, QSWR	:: BYPASS DRIVE STATUS TYPEOUT ?
2295	004432	001044				BNE	START7	:: BR IF YES
2296	004434	104401	027236		95:	TYPE	DRSTAT	:: 'DRIVE STATUS'
2297	004440	005001				CLR	R1	:: CLEAR TABLE POINTER
2298	004442	012702	000010			MOV	#B, .R2	:: COUNTER
2299	004446				35:			
2300	004446	010146				MOV	R1, - SF,	:: SAVE R1 FOR TYPEOUT
2301								:: TYPE DRIVE NUMBER
2302	004450	104403				TIPOS		:: GC TYPE--OCTAL ASCII
2303	004452	CC1				BITE	1	:: TYPE 1 DIGITS

2304	004453	000			.BYTE	0	:: SUPPRESS LEADING ZEROS
2305	004454	105761	001370		TSTB	DRVSTA(R1)	: CHECK DRIVE'S STATUS
2306	004460	001420			BEQ	7\$: DRIVE IS OFFLINE OR NON-EXISTENT
2307	004462	100403			BMI	4\$: DRIVE IS UNSAFE
2308	004464	104401	027262		TYPE	ONLINE	: 'ONLINE'
2309	004470	000402			BR	5\$	
2310	004472	104401	027276		TYPE	UNSAFE	: 'UNSAFE'
2311	004476	136137	001360	001304	4\$: BITB	ATABIT(R1), DRVTYPE	: ;SEE WHICH DRIVE TYPE
2312	004504	001003			BNE	6\$: BR IF RPO3
2313	004506	104401	027327		TYPE	RPO2	: 'RPO2'
2314	004512	000405			BR	8\$	
2315	004514	104401	027336		6\$: TYPE	RPO3	: 'RPO3'
2316	004520	000402			BR	8\$	
2317	004522	104401	027312		7\$: TYPE	.OFFLIN	: 'OFFLINE'
2318	004526	104401	001173		8\$: TYPE	\$SCRLF	: CR-LF
2319	004532	005201			INC	R1	: INCREMENT TABLE POINTER
2320	004534	005302			DEC	R2	: DECREMENT THE COUNTER
2321	004536	001343			BNE	3\$: CONTINUE
2322	004540	104401	001173		TYPE	\$SCRLF	: CR-LF
2323	004544	005737	001206		START7: TST	CNTRLC	: CONVERSATION MODE ?
2324	004550	001403			BEQ	1\$: NO--BRANCH
2325	004552	004737	026314		JSR	PC,ENTPRM	: YES--GET PARAMETERS
2326	004556	000416			BR	5\$: GO TYPE DRIVES TO BE TESTED
2327	004560	005037	001312		1\$: CLR	DRVSEL	: NO DRIVES SELECTED
2328	004564	005000			CLR	R0	: DETERMINE THE DRIVES THAT
2329	004566	012701	000001		MOV	#1,R1	: ARE AVAILABLE FOR TESTING
2330	004572	105760	001370		2\$: TSTB	DRVSTA(R0)	: DRIVE ONLINE ?
2331	004576	003403			J\$	3\$: BR IF NOT
2332	004600	156037	001360	001312	3\$: BISB	ATABIT(R0), DRVSEL	: ;SET SELECTION BIT FOR DRIVE
2333	004606	005200			INC	R0	: INCREMENT DRIVE ADDRESS
2334	004610	106301			ASLB	R1	: COUNT
2335	004612	001367			BNE	2\$: BR IF NOT ALL DRIVES CHECKED
2336	004614	104401	027345		5\$: TYPE	DRVST	: 'DRIVES TO BE TESTED'
2337	004620	005037	017004		\$ENDCT		: DETERMINE PASSES TO MAKE AND
2338	004624	005000			CLR	R0	: THE DRIVES TO BE TESTED
2339	004626	013701	001312		MOV	DRVSEL,R1	: ANY DRIVES SELECTED?
2340	004632	001004			BNE	6\$: YES--BRANCH
2341	004634	104401	027372		TYPE	NONE	: 'NONE'
2342	004640	000137	016616		J\$SEOP		: GO TO END OF PROGRAM
2343	004644	005737	001206		6\$: TST	CNTRLC	: CONVERSATION MODE START ?
2344	004650	001011			BNE	7\$: BR IF NOT
2345	004652	005737	000042		TST	42	: UNDER MONITOR CONTROL ?
2346	004656	001406			BEQ	7\$: BR IF NOT
2347	004660	122737	000003	000041	CMPB	#3,41	: LOADED BY RPO2/RPO3 ?
2348	00466	001002			BNE	7\$: BR IF NOT
2349	004670	006201			ASR	R1	: EXCLUDE DRIVE 0 FROM TESTING
2350	004672	000413			BR	8\$: CHECK ON THE OTHER DRIVES
2351	004674	006201			7\$: ASR	R1	: REPORT THE DRIVES TO BE TESTED
2352	004676	103011			BCC	8\$: BR IF DRIVE IS NOT SELECTED
2353	004700	005237	017004		INC	\$ENDCT	: GIVE THIS DRIVE A PASS
2354	004704	010046			MOV	R0,-(SP	: ;SAVE R0 FOR TYPEOUT
2355							: ;TYPE THE DRIVE NUMBER
2356	004706	104402			TYPOS		: ;GO TYPE--OCTAL ASCII
2357	004710	001			.BYTE	1	: ;TYPE 1 DIGIT(S)
2358	004711	000			.BYTE	0	: ;SUPPRESS LEADING ZEROS
2359	004712	0057C1			TET	R1	: ;MORE DRIVES?

```

2360 004714 001404          BEQ      9$      ;NO--BRANCH
2361 004716 104401 027377  TYPE     COMMA
2362 004722 005200          INC      AO      ;INCREMENT DRIVE NUMBER
2363 004724 000763          BR       7$      ;CONTINUE
2364 004726 013737 017004 016776 9$:      MOV     $ENDCT,$EOPCT ;SETUP TEST COUNT
2365 004734 005037 001206          CLR     CNTRLC   ;CLEAR CONVERSATION MODE FLAG
2366 004740 005037 001176          RSTART1: CLR    CHKDRV ;INITIALIZE THE CHECK DRIVE KEY
2367 004744 012737 000001 001306 MOV     #1,DRVMSK ;START TO CHECK DESIRED DRIVES
2368 004752 033737 001306 001312 RSTART2: BIT   DRVMSK,DRVSEL ;IS THIS DRIVE SELECTED?
2369 004760 001011          RSTART3: BNE   DRVOK ;YES--GO CHECK IF DRIVE IS READY FOR TESTING
2370 004762 005237 001176          RESTART: INC   CHKDRV ;MOVE TO NEXT DRIVE NUMBER
2371 004766 106337 001306          ASLB   DRVMSK   ;POSITION THE MASK
2372 004772 103367          BCC     RSTART2 ;BR IF MORE DRIVES
2373 004774 043737 001310 001312 BIC     DRVBAD,DRVSEL ;CLEAR SELECTION BITS FOR ANY OFFLINE/UNSAFE
2374                                DRIVES
2375 005002 000756          BR       RSTART1 ;CONTINUE WITH CYCLE
2376 005004 113737 001176 002500 DRVOK:  MOVB  CHKDRV,DPB ;PICKUP THE DRIVE NUMBER
2377 005012 104401 027401          TYPE   TSTING   ;TESTING WITH DRIVE
2378 005016 013746 001176          MOV     CHKDRV,-(SP) ;SAVE CHKDRV FOR TYPEOUT
2379                                ;TYPE THE DRIVE NUMBER
2380 005022 104403          TYPOS   ;GO TYPE--OCTAL ASCII
2381 005024 001          .BYTE  1      ;TYPE 1 DIGIT(S)
2382 005025 000          .BYTE  0      ;SUPPRESS LEADING ZEROS
2383 005026 104401 001173          TYPE   $CRLF   ;CR-LF
2384 005032 013704 001324          MOV     RPADR,R4 ;RP11 ADDRESS
2385 005036 004737 025654          JSR     PC,CLRP ;CLEAR THE RP11
2386 005042 112737 000015 002502 MOVB   #HOMSEK,DPB+2 ;HOME SEEK COMMAND
2387 005050 004737 022770          JSR     PC,RP11 ;START THE COMMAND
2388 005054 004707 023546          JSR     PC,DRVROY ;WAIT FOR THE DRIVE TO FINISH
2389 005060 000137 005064          JMP     TST0    ;START THE PROGRAM
2390
2391          .SBTTL  **** TESTS ****
2392
2393          ;*****
2394          ;*TEST 0          TEST 'CLEAR' TERMINATION
2395          ;*****
2396          ;*VERIFY THAT 'CLEAR' TERMINATES AN OPERATION AND CAUSES CONTROLLER
2397          ;*READY TO SET.
2398          ;*****
2399
2400          ;*****
2401          TST0:
2402          BIT     BITS+($TN*2),TSTNMS ;IS THIS TEST SELECTED
2403          BNE    .+6          ;BR IF YES
2404          JMP     TST1        ;GO TO THE NEXT TEST
2405          MOV     #TEST0,$LPERR ;SETUP THE ERROR LOOP ADDRESS
2406          MOV     #TST0,$LPADR ;SETUP THE SCOPE LOOP ADDRESS
2407          MOV     #0,$TSTNM   ;SETUP TEST NUMBER AND
2408                                ;CLEAR THE ERROR FLAG ($ERFLG)
2409          MCV     $TSTNM,$DISPLAY ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
2410          MOV     #EXIT0,BYPASS ;SETUP BYPASS ADDRESS
2411          MOV     RPADR,R4    ;RP11E BUS ADDRESS
2412          CLRB   DPB+1       ;CLEAR ANY EXTENDED MEMORY BITS
2413          CLRB   DPB+3       ;CLEAR 'MODE' & 'HDR' BITS
2414          MOV     #100,$TIMES ;DO 100 ITERATIONS
2415          MOV     #STACK,$SP  ;SETUP THE STACK POINTER

```

```

2416 005164 004737 025654 JSR PC,CLRP ;CLEAR THE RP11
2417 005170 012737 043656 002506 MOV #BUFFER,$BUF ;LOAD BUFFER ADDRESS
2418 005176 012737 000001 002504 MOV #1,$WC ;SETUP WORD COUNT FOR 1 WORD
2419 005204 005037 002510 CLR $CYL ;CYLINDER 0
2420 005210 005037 002512 CLR $SEC ;TRACK & SECTOR 0
2421 005214 012737 000005 002502 MOVB #ROSEK,DPB+2 ;READ COMMAND
2422 005222 004737 022770 JSR PC,RP11 ;ISSUE THE COMMAND AND RETURN
2423 005226 032764 000200 000004 BIT #RDY,$RPCS(R4) ;IS THE CONTROLLER READY?
2424 005234 001404 BEQ $S ;BR IF NOT
2425 005236 004737 023240 JSR PC,SAVRP ;STORE THE REGISTERS
2426 005242 104002 ERROR 2 ;CAN'T START READ COMMAND
2427 005244 000420 BR ;EXIT FROM TEST
2428 005246 012764 000001 000004 1$: MOV #CLEAR,$RPCS(R4) ;ISSUE THE CONTROLLER CLEAR
2429 005254 012764 000001 000004 MOV #CLEAR,$RPCS(R4) ;CLEAR THE CONTROLLER AGAIN
2430 005262 004737 023240 JSR PC,SAVRP ;STORE THE REGISTERS
2431 005266 032737 000200 001340 BIT #RDY,$RPCS ;IS THE CONTROLLER READY?
2432 005274 001004 BNE EXITO ;BR IF IT IS
2433 005276 104003 ERROR 3 ;'READY' DIDN'T SET AFTER ISSUING 'CLEAR'
2434 ; DURING READ COMMAND
2435 ; FORCE THE CONTROLLER READY
2436 005300 000005 RESET ;
2437 005302 004737 020416 JSR PC,$TKINT ;SETUP THE TTY KEYBOARD
2438 005306 000004 EXITO: SCOPE ;LOOP?

```

: *TEST 1 WRITE CHECK TEST

: *THIS TEST VERIFIES THE WRITE CHECK LOGIC AND TO VERIFY THAT A WRITE
: *CHECK ERROR CAN BE DETECTED. 'FLOATING ONES' AND 'FLOATING ZEROS'
: *PATTERNS ARE USED TO TEST THE WRITE CHECK COMPARE LOGIC.

↑ST1:

```

2448 005310 033737 001402 001276 BIT BITS+<$TN*2>,$TSTNMS ;IS THIS TEST SELECTED
2449 005310 001002 BNE +6 ;BR IF YES
2450 005316 000137 005724 JMP $TST2 ;GO TO THE NEXT TEST
2451 005320 012737 005404 001110 MOV #TEST1,$LPERR ;SETUP THE ERROR LOOP ADDRESS
2452 005332 012737 005310 001106 MOV #TST1,$LPADR ;SETUP THE SCOPE LOOP ADDRESS
2453 005340 012737 000001 001102 MOV #1,$TSTNM ;SETUP TEST NUMBER AND
2454 ; CLEAR THE ERROR FLAG ($ERFLG)
2455 005346 013777 001102 173566 MOV $TSTNM,$DISPLAY ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
2456 005354 012737 005722 001200 MOV #EXIT1,BYPASS ;SETUP BYPASS ADDRESS
2457 005362 013704 001324 MOV RPADR,$R4 ;RP11E BUS ADDRESS
2458 005366 105037 002501 CLRB DPB+1 ;CLEAR ANY EXTENDED MEMORY BITS
2459 005372 105037 002503 CLRB DPB+3 ;CLEAR 'MODE' & 'HDR' BITS
2460 005376 012737 000144 001162 MOV #100,$TIMES ;DO 100 ITERATIONS
2461 005404 012706 001100 TEST1: MOV #STACK,$SP ;SETUP THE STACK POINTER
2462 005410 005037 002512 CLR $SEC ;CLEAR TRACK & SECTOR ADDRESS
2463 005414 005037 002510 CLR $CYL ;CLEAR CYLINDER ADDRESS
2464 005420 012737 000400 002504 MOV #256,$WC ;SET WORD COUNT TO 1 SECTOR
2465 005426 012737 000011 001204 MOV #9,$PATNUM ;STARTING PATTERN NUMBER
2466 005434 004737 024432 JSR PC,$SETBUF ;FILL THE BUFFER WITH THE PATTERN
2467 005440 012737 043656 002506 MOV #BUFFER,$BUF ;SETUP OUTPUT BUFFER
2468 005446 012737 005446 001110 1$: MOV #IS,$LPERR ;SETUP LOOP ON ERROR ADDRESS
2469 005454 004737 025654 JSR PC,CLRP ;CLEAR THE RP11
2470 005460 012737 000003 002502 MOV #WATSEK,DPB+2 ;LOAD COMMAND

```


T1 WRITE CHECK TEST

```

2472 005466 004737 022770 JSR PC,RP11 ;WRITE THE SECTOR WITH THE TEST PATTERN
2473 005472 004737 023664 JSR PC,CONRDY ;WAIT FOR THE OPERATION TO COMPLETE
2474 005476 004737 023240 JSR PC,SAVRP ;SAVE THE REGISTERS
2475 005502 032737 100000 001340 BIT #ERR,$RPCS ;DID AN ERROR OCCUR ?
2476 005510 001401 BEQ 3$ ;BR IF NOT
2477 005512 104004 ERROR 4 ;ERROR OCCURED WRITING WRITE CHECK TEST PATTERN
2478 005514 012737 005514 001110 3$: MOV #3$,SLPERR ;CHANGE LOOP ON ERROR ADDRESS
2479 005522 004737 025654 JSR PC,CLRP ;CLEAR THE RP11
2480 005526 012737 000007 002502 MOV #WATCHK,DPB+2 ;LOAD WRITE CHECK COMMAND
2481 005534 004737 022770 JSR PC,RP11 ;CHECK THE SECTOR JUST WRITTEN
2482 005540 004737 023664 JSR PC,CONRDY ;WAIT FOR WRITE CHECK TO COMPLETE
2483 005544 004737 023240 JSR PC,SAVRP ;STORE THE REGISTERS
2484 005550 032737 100000 001340 BIT #ERR,$RPCS ;CHECK OK ?
2485 005556 001403 BEQ 4$ ;BR IF OK
2486 005560 104005 ERROR 5 ;ERROR ATTEMPTING TO CHECK TEST PATTERN
2487 005562 004737 025654 JSR PC,CLRP ;CLEAR THE RP11
2488 005566 005001 4$: CLR R1 ;SETUP TO CLEAR BUFFER
2489 005570 005002 CLR R2 ;BUFFER INDEX
2490 005572 023727 001204 000011 CMP PATNUM,#9. ;'FLOATING 1' PATTERN ?
2491 005600 001002 BNE 5$ ;BRANCH IF NOT
2492 005602 012701 MOV #-1,R1 ;FILL BUFFER WITH ONES
2493 005606 010162 043656 5$: MOV R1,BUFFER(R2) ;FILL BUFFER WITH PATTERN IN R1
2494 005612 022702 000400 CMP #256.,R2 ;SEE IF FINISHED
2495 005616 001403 BEQ 6$ ;BR IF FINISHED
2496 005620 062702 000002 ADD #2,R2 ;INCREMENT THE INDEX
2497 005624 000770 BR 5$ ;CONTINUE
2498 005626 012737 005626 001110 6$: MOV #6$,SLPERR ;CHANGE THE LOOP ON ERROR ADDRESS
2499 005634 004737 025654 JSR PC,CLRP ;CLEAR THE RP11
2500 005640 004737 022770 JSR PC,RP11 ;CHECK THE SECTOR AGAIN, ERROR SHOULD OCCUR
2501 005644 004737 023664 JSR PC,CONRDY ;WAIT FOR THE WRITE CHECK TO COMPLETE
2502 005650 004737 023240 JSR PC,SAVRP ;STORE THE REGISTERS
2503 005654 032737 000010 001336 BIT #BIT03,$RPER ;DID WRITE CHECK ERROR SET ?
2504 005662 001002 BNE 7$ ;BRANCH IF YES
2505 005664 104006 ERROR 6 ;WRITE CHECK ERROR DID NOT SET
2506 005666 000405 BR 8$ ;BYPASS REST OF TEST
2507 005670 032737 100000 001340 7$: BIT #ERR,$RPCS ;DID 'ERR' SET ?
2508 005676 001001 BNE 8$ ;BR IF IT SET
2509 005700 104106 ERROR 106 ;'ERR' DIDN'T SET WITH 'WCE'
2510 005702 023727 001204 000012 8$: CMP PATNUM,#10. ;FLOATING A ZERO ?
2511 005710 001404 BEQ EXIT1 ;BR IF ZERO
2512 005712 012737 000012 001204 MOV #10.,PATNUM ;'FLOATING ZERO' PATTERN
2513 005720 000652 BR 1$ ;DO THE TEST AGAIN
2514 005722 000004 EXIT1: SCOPE ;LOOP ?

```

```

*****
;TEST 2 PARTIAL SECTOR WRITE TEST

```

```

; *CHECK THE ABILITY OF THE RP11E TO CLEAR THE REMAINDER OF A SECTOR
; *ON A PARTIAL WRITE OPERATION. A SECTOR OF ALL ONES IS WRITTEN AND
; *THEN A TWO WORD WRITE OPERATION IS PERFORMED. THE SECTOR IS THEN
; *READ BACK AND VERIFIED. THE FIRST TWO WORDS SHOULD BE ONES AND
; *THE REST SHOULD BE ZEROS.

```

```

*****
;S*2:

```

005724

2528	005724	033737	001404	001276	BIT	#STN*2, TSTNMS	IS THIS TEST SELECTED	
2529	005732	001002			BNE	+6	BR IF YES	
2530	005734	000137	006402		JMP	TST3	GO TO THE NEXT TEST	
2531	005740	012737	006020	001110	MOV	#TEST2, \$LPERR	SETUP THE ERROR LOOP ADDRESS	
2532	005746	012737	005724	001106	MOV	#TST2, \$LPADR	SETUP THE SCOPE LOOP ADDRESS	
2533	005754	012737	000002	001102	MOV	#2, \$TSTNM	SETUP TEST NUMBER AND	
2534							CLEAR THE ERROR FLAG (\$ERFLG)	
2535	005762	013777	001102	173152	MOV	\$TSTNM, @DISPLAY	LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER	
2536	005770	012737	005400	001200	MOV	#EXIT2, BYPASS	SETUP BYPASS ADDRESS	
2537	005776	013704	001324		MOV	RPADR, R4	RP11E BUS ADDRESS	
2538	006002	105037	002501		CLRB	DPB+1	CLEAR ANY EXTENDED MEMORY BITS	
2539	006006	105037	002503		CLRB	DPB+3	CLEAR 'MODE' & 'HDR' BITS	
2540	006012	012737	000144	001162	MOV	#100, \$TIMES	DO 100 ITERATIONS	
2541	006020	012706	001100		MOV	#STACK, SP	SETUP THE STACK POINTER	
2542	006024	012737	006020	001110	MOV	#TEST2, \$LPERR	SETUP INITIAL LOOP ON ERROR ADDRESS	
2543	006032	004737	025654		JSR	PC, CLRP	CLEAR THE RP11	
2544	006036	012737	043656	002506	MOV	#BUFFER, \$BUF	LOAD BUFFER ADDRESS	
2545	006044	012737	000400	002504	MOV	#256, \$WC	WORD COUNT - 1 SECTOR	
2546	006052	005037	002512		CLR	\$SEC	CLEAR SECTOR/TRACK ADDRESS	
2547	006056	005037	002510		CLR	\$CYL	CLEAR CYLINDER ADDRESS	
2548	006062	012737	000015	001204	MOV	#13, \$PATNUM	ONE'S PATTERN NUMBER	
2549	006070	004737	024432		JSR	PC, SETBUF	LOAD THE PATTERN INTO THE BUFFER	
2550	006074	012737	000003	002502	MOV	#WATSEK, DPB+2	WRITE COMMAND	
2551	006102	004737	022770		JSR	PC, RP11	WRITE THE SECTOR	
2552	006106	004737	023664		JSR	PC, CONRDY	WAIT FOR THE WRITE TO COMPLETE	
2553	006112	004737	023240		JSR	PC, SAVRP	STORE THE REGISTERS	
2554	006116	032737	100000	001340	BIT	#ERR, \$RPCS	DID AN ERROR OCCUR ?	
2555	006124	001402			BEQ	15	BR IF NOT	
2556	006126	104004			ERROR	4	ERROR OCCURED DURING WRITE	
2557	006130	000523			BR	EXIT2	BYPASS REST OF TEST	
2558	006132	012737	006132	001110	MOV	#15, \$LPERR	CHANGE LOOP ON ERROR ADDRESS	
2559	006140	004737	025654		JSR	PC, CLRP	CLEAR THE RP11	
2560	006144	004737	024432		JSR	PC, SETBUF	LOAD ONE'S PATTERN INTO THE BUFFER	
2561	006150	012737	000002	002504	MOV	#2, \$WC	2 WORD TRANSFER	
2562	006156	112737	000003	002502	MOVB	#WATSEK, DPB+2	CHANGE COMMAND TO WRITE	
2563	006164	004737	022770		JSR	PC, RP11	WRITE	
2564	006170	004737	023664		JSR	PC, CONRDY	WAIT FOR CONTROLLER READY	
2565	006174	004737	023240		JSR	PC, SAVRP	STORE THE REGISTERS	
2566	006200	032737	100000	001340	BIT	#ERR, \$RPCS	ERROR ?	
2567	006206	001402			BEQ	25	BR IF NOT	
2568	006210	104007			ERROR	7	ERROR WRITING THE PARTIAL SECTOR	
2569	006212	000472			BR	EXIT2	BYPASS THE REST OF THE TEST	
2570	006214	012737	006214	001110	MOV	#25, \$LPERR	CHANGE THE ERROR LOOP ADDRESS	
2571	006222	004737	025654		JSR	PC, CLRP	CLEAR THE RP11	
2572	006226	012737	000400	002504	MOV	#256, \$WC	CHANGE THE WORD COUNT	
2573	006234	012737	000005	002502	MOV	#RDSEK, DPB+2	READ COMMAND	
2574	006242	004737	022770		JSR	PC, RP11	READ THE SECTOR	
2575	006246	004737	023664		JSR	PC, CONRDY	WAIT FOR CONTROLLER READY	
2576	006252	004737	023240		JSR	PC, SAVRP	STORE THE REGISTERS	
2577	006256	032737	100000	001340	BIT	#ERR, \$RPCS	DID AN ERROR OCCUR ?	
2578	006264	001402			BEQ	35	BR IF NOT	
2579	006266	104017			ERROR	17	ERROR OCCURED READING SECTOR	
2580	006270	000443			BR	EXIT2	BYPASS REST OF TEST	
2581	006272	022737	177777	043656	3\$:	CMP	#177777, BUFFER	COMPARE FIRST WORD SHOULD BE ONES
2582	006300	001004			BNE	45	BR IF NOT	
2583	006302	022737	177777	043660	CMP	#177777, BUFFER+2	CHECK THE SECOND WORD	

```

2584 006310 001407 BEQ 5$ ;BR IF OK
2585 006312 012737 177777 001124 4$: MOV #177777,$GDDAT ;EXPECTED DATA
2586 006320 013737 043656 001126 MOV BUFFER,$BDDAT ;ACTUAL DATA
2587 006326 104011 ERROR 11 ;DATA COMPARE ERROR IN FIRST 2 WORDS
2588 006330 012700 043662 5$: MOV #BUFFER+4,R0 ;STARTING ADDRESS OF LAST PART OF THE BUFFER
2589 006334 012701 000376 MOV #254,R1 ;WORD COUNT
2590 006340 005720 6$: TST (R0)+ ;REMAINDER OF SECTOR SHOULD BE ZEROS
2591 006342 001003 BNE 7$ ;BRANCH IF NOT
2592 006344 005301 DEC R1 ;DECREMENT THE CUNT
2593 006346 001374 BNE 6$ ;BR IF NOT FINISHED
2594 006350 000413 BR EXIT2 ;EXIT
2595 006352 016037 177776 001126 7$: MOV -2(R0),$BDDAT ;INCORRECT DATA
2596 006360 005037 001124 CLR $GDDAT ;EXPECTED DATA (ZEROS)
2597 006364 010037 001122 MOV R0,$BDADR ;ADDRESS OF INCORRECT CHARACTER
2598 006370 162737 000002 001122 SUB #2,$BDADR ;DECREMENT THE ADDRESS
2599 006376 104012 ERROR 12 ;DATA FOUND IN AREA OF SECTOR
;WHICH SHOULD HAVE BEEN CLEARED
;BY A ONE WORD WRITE

```

2601 2602 006400 000004 EXIT2: SCOPE

2606 ;*TEST 3 'EOP' TEST

```

2608 ;*TEST THAT 'EOP' SETS WHEN THE CONTROLLER TRIES TO WRITE BEYOND THE
2609 ;*LIMITS OF THE DRIVE. THE FIRST SECTOR OF THE PACK IS WRITTEN WITH
2610 ;*ZEROS. THEN A TWO SECTOR WRITE OF ALL ONE'S IS ISSUED FOR THE MAXIMUM
2611 ;*CYLINDER, HEAD 19, SECTOR 9. 'EOP' AND THE ERROR BITS IN 'RPCS' SHOULD
2612 ;*BE SET. THE FIRST SECTOR OF THE PACK IS CHECKED TO MAKE SURE THAT IS
2613 ;*STILL CONTAINS ZEROS.

```

```

2616 006402 †ST3:
2617 006402 033737 001406 001276 BIT BITS+(<STN*2),TSTNMS ;IS THIS TEST SELECTED
2618 006410 001002 BNE +6 ;BR IF YES
2619 006412 000137 007060 JMP TST4 ;GO TO THE NEXT TEST
2620 006416 012737 006476 001110 MOV #TEST3,$LPERR ;SETUP THE ERROR LOOP ADDRESS
2621 006424 012737 006402 001106 MOV #TST3,$LPADR ;SETUP THE SCOPE LOOP ADDRESS
2622 006432 012737 000003 001102 MOV #3,$TSTNM ;SETUP TEST NUMBER AND
2623 ;CLEAR THE ERROR FLAG ($ERFLG)
2624 006440 013777 001102 172474 MOV $TSTNM,$DISPLAY ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
2625 006446 012737 007056 001200 MOV #EXIT3,BYPASS ;SETUP BYPASS ADDRESS
2626 006454 013704 001324 MOV RPADR,R4 ;RP11E BUS ADDRESS
2627 006460 105037 002501 CLR DPB+1 ;CLEAR ANY EXTENDED MEMORY BITS
2628 006464 105037 002503 CLR DPB+3 ;CLEAR 'MODE' & 'HDR' BITS
2629 006470 012737 000031 001162 MOV #25,$TIMES ;DO 25 ITERATIONS
2630 006476 012706 001100 TEST3: MOV #STACK,SP ;SETUP THE STACK POINTER
2631 006502 012737 006476 001110 MOV #TEST3,$LPERR ;INITIAL LOOP ON ERROR ADDRESS
2632 006510 004737 025654 JSR PC,CLRP ;CLEAR THE RP11
2633 006514 012737 000400 002504 MOV #256,$WC ;SET WORD COUNT TO 1 SECTOR
2634 006522 012737 043656 002506 MOV #BUFFER,$BUF ;SETUP OUTPUT BUFFER
2635 006530 005037 001204 CLR PATNUM ;SETUP FOR PATTERN ZERO
2636 006534 004737 024432 JSR PC,SETBUF ;LOAD ZEROS INTO THE BUFFER
2637 006540 005037 002512 CLR $SEC ;SET SECTOR & TRACK TO ZERO
2638 006544 005037 002510 CLR $CYL ;SET CYLINDER TO ZERO
2639 006550 112737 000003 002502 MOVB #WRTSEK,DPB+2 ;WRITE COMMAND

```

```

26800 006556 004737 022770
26801 006557 004737 022770
26802 006558 004737 022770
26803 006559 004737 022770
26804 006560 004737 022770
26805 006561 004737 022770
26806 006562 004737 022770
26807 006563 004737 022770
26808 006564 004737 022770
26809 006565 004737 022770
26810 006566 004737 022770
26811 006567 004737 022770
26812 006568 004737 022770
26813 006569 004737 022770
26814 006570 004737 022770
26815 006571 004737 022770
26816 006572 004737 022770
26817 006573 004737 022770
26818 006574 004737 022770
26819 006575 004737 022770
26820 006576 004737 022770
26821 006577 004737 022770
26822 006578 004737 022770
26823 006579 004737 022770
26824 006580 004737 022770
26825 006581 004737 022770
26826 006582 004737 022770
26827 006583 004737 022770
26828 006584 004737 022770
26829 006585 004737 022770
26830 006586 004737 022770
26831 006587 004737 022770
26832 006588 004737 022770
26833 006589 004737 022770
26834 006590 004737 022770
26835 006591 004737 022770
26836 006592 004737 022770
26837 006593 004737 022770
26838 006594 004737 022770
26839 006595 004737 022770
26840 006596 004737 022770
26841 006597 004737 022770
26842 006598 004737 022770
26843 006599 004737 022770
26844 006600 004737 022770
26845 006601 004737 022770
26846 006602 004737 022770
26847 006603 004737 022770
26848 006604 004737 022770
26849 006605 004737 022770
26850 006606 004737 022770
26851 006607 004737 022770
26852 006608 004737 022770
26853 006609 004737 022770
26854 006610 004737 022770
26855 006611 004737 022770
26856 006612 004737 022770
26857 006613 004737 022770
26858 006614 004737 022770
26859 006615 004737 022770
26860 006616 004737 022770
26861 006617 004737 022770
26862 006618 004737 022770
26863 006619 004737 022770
26864 006620 004737 022770
26865 006621 004737 022770
26866 006622 004737 022770
26867 006623 004737 022770
26868 006624 004737 022770
26869 006625 004737 022770
26870 006626 004737 022770
26871 006627 004737 022770
26872 006628 004737 022770
26873 006629 004737 022770
26874 006630 004737 022770
26875 006631 004737 022770
26876 006632 004737 022770
26877 006633 004737 022770
26878 006634 004737 022770
26879 006635 004737 022770
26880 006636 004737 022770
26881 006637 004737 022770
26882 006638 004737 022770
26883 006639 004737 022770
26884 006640 004737 022770
26885 006641 004737 022770
26886 006642 004737 022770
26887 006643 004737 022770
26888 006644 004737 022770
26889 006645 004737 022770
26890 006646 004737 022770
26891 006647 004737 022770
26892 006648 004737 022770
26893 006649 004737 022770
26894 006650 004737 022770
26895 006651 004737 022770
26896 006652 004737 022770
26897 006653 004737 022770
26898 006654 004737 022770
26899 006655 004737 022770
26900 006656 004737 022770
26901 006657 004737 022770
26902 006658 004737 022770
26903 006659 004737 022770
26904 006660 004737 022770
26905 006661 004737 022770
26906 006662 004737 022770
26907 006663 004737 022770
26908 006664 004737 022770
26909 006665 004737 022770
26910 006666 004737 022770
26911 006667 004737 022770
26912 006668 004737 022770
26913 006669 004737 022770
26914 006670 004737 022770
26915 006671 004737 022770
26916 006672 004737 022770
26917 006673 004737 022770
26918 006674 004737 022770
26919 006675 004737 022770
26920 006676 004737 022770
26921 006677 004737 022770
26922 006678 004737 022770
26923 006679 004737 022770
26924 006680 004737 022770
26925 006681 004737 022770
26926 006682 004737 022770
26927 006683 004737 022770
26928 006684 004737 022770
26929 006685 004737 022770
26930 006686 004737 022770
26931 006687 004737 022770
26932 006688 004737 022770
26933 006689 004737 022770
26934 006690 004737 022770
26935 006691 004737 022770
26936 006692 004737 022770
26937 006693 004737 022770
26938 006694 004737 022770
26939 006695 004737 022770
26940 006696 004737 022770
26941 006697 004737 022770
26942 006698 004737 022770
26943 006699 004737 022770
26944 006700 004737 022770
26945 006701 004737 022770
26946 006702 004737 022770
26947 006703 004737 022770
26948 006704 004737 022770
26949 006705 004737 022770
26950 006706 004737 022770
26951 006707 004737 022770
26952 006708 004737 022770
26953 006709 004737 022770
26954 006710 004737 022770
26955 006711 004737 022770
26956 006712 004737 022770
26957 006713 004737 022770
26958 006714 004737 022770
26959 006715 004737 022770
26960 006716 004737 022770
26961 006717 004737 022770
26962 006718 004737 022770
26963 006719 004737 022770
26964 006720 004737 022770
26965 006721 004737 022770
26966 006722 004737 022770
26967 006723 004737 022770
26968 006724 004737 022770
26969 006725 004737 022770
26970 006726 004737 022770
26971 006727 004737 022770
26972 006728 004737 022770
26973 006729 004737 022770
26974 006730 004737 022770
26975 006731 004737 022770
26976 006732 004737 022770
26977 006733 004737 022770
26978 006734 004737 022770
26979 006735 004737 022770
26980 006736 004737 022770
26981 006737 004737 022770
26982 006738 004737 022770
26983 006739 004737 022770
26984 006740 004737 022770
26985 006741 004737 022770
26986 006742 004737 022770
26987 006743 004737 022770
26988 006744 004737 022770
26989 006745 004737 022770
26990 006746 004737 022770
26991 006747 004737 022770
26992 006748 004737 022770
26993 006749 004737 022770
26994 006750 004737 022770
26995 006751 004737 022770
26996 006752 004737 022770
26997 006753 004737 022770
26998 006754 004737 022770
26999 006755 004737 022770
27000 006756 004737 022770

```

```

JSR PC,RP11
JSR PC,CONADY
JSR PC,SAVRP
BRAR,SRPCS
BR 18
EXIT3
JSR B18,SLPERR
PC,CLRP
JSR B13,PATNUM
PC,SETBLF
JSR B13,SEC
MAXCYL,SCYL
B19,SEC
JSR B19,STAK
PC,RP11
JSR PC,CONADY
JSR PC,SAVRP
BIT B101,SRPER
BR 28
JSR B101,SRPER
BR 38
JSR B101,SRPER
BR 48
JSR B101,SRPER
BR 58
JSR B101,SRPER
BR 68
EXIT3
MOV BUFFER,SBODAT
TST SBODAT
EXIT3
CLR SBODAT
ERROR 20

```

```

WRITE ZEROS IN CO. TO .50
WAIT FOR THE ORDER TO COMPLETE
STORE THE REGISTERS
DID AN ERROR OCCUR?
BRANCH IF NOT
ERROR OCCURED DURING SETUP
BYPASS REST OF TEST
CHANGE LOOP ON ERROR ADDRESS
CLEAR THE RP11
ONE'S PATTERN INDEX
LOAD THE PATTERN INTO THE BUFFER
CHANGE WORD COUNT TO 2 SECTORS
SELECT THE DRIVE'S MAXIMUM CYLINDER
SELECT SECTOR 9
START THE READ
WAIT FOR CONTROLLER READY
STORE THE RP11 REGISTERS
DID EOP ERROR SET?
BRANCH IF SET
'EOP' DIDN'T SET
CHECK RP11
DID THE ERROR FLAG SET?
BRANCH IF SET
ERROR DID NOT SET AFTER GENERATING 'EOP'
SEE IF THE CYLINDER ADDRESS REGISTER WAS CLEARED
BR IF CLEARED
'RP11' NOT CLEARED BY 'EOP'
SUCA STILL EQUAL TO THE MAXIMUM CLINDER
BR IF IT IS
SUCA NOT CORRECT AFTER EOP ERROR
CHANGE LOOP ON ERROR ADDRESS
CLEAR THE RP11
WORD COUNT
CLEAR SECTOR/TRACK ADDRESS
CLEAR THE CYLINDER ADDRESS
CHANGE COMMAND TO READ
EXECUTE THE COMMAND
WAIT FOR CONTROLLER READY
STORE THE REGISTERS
WERE THERE ANY ERRORS?
BRANCH IF NOT
ERROR ENCOUNTERED ON 2 WORD REAC
OF FIRST SECTOR ON THE PACK
GET FIRST WORD OF BUFFER
DOES 1ST SECTOR STILL CONTAIN ZEROS?
BRANCH IF YES
ZEROS WERE EXPECTED
CONTENTS OF THE FIRST SECTOR OF THE
PACK CHANGED AFTER FORCING EOP

```

```

*****
; *TEST 4 TEST 'PROG' ERROR

```

VERIFY THAT THE RP11E GENERATES A 'PROG' ERROR IF A COMMAND IS ISSUED WHILE THE CONTROLLER IS BUSY.

†ST4:

2700	007060	033737	001410	001276	BIT	BITS+(\$TN*2),TSTNMS	: IS THIS TEST SELECTED
2701	007060	001002			BNE	+6	: BR IF YES
2702	007066	000137	007310		JMP	†ST5	: GO TO THE NEXT TEST
2703	007070	012737	007154	001110	MOV	†TEST4,\$LPERR	: SETUP THE ERROR LOOP ADDRESS
2704	007074	012737	007060	001106	MOV	†ST4,\$LPADR	: SETUP THE SCOPE LOOP ADDRESS
2705	007102	012737	000004	001102	MOV	†4,†STNM	: SETUP TEST NUMBER AND
2706							: CLEAR THE ERROR FLAG (SERFLG)
2707							: LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
2708	007116	013777	001102	172016	MOV	†STNM,†DISPLA	: SETUP BYPASS ADDRESS
2709	007124	012737	007306	001200	MOV	†EXIT4,BYPASS	: RP11E BUS ADDRESS
2710	007132	013704	001324		MOV	†RPAOR,R4	: CLEAR ANY EXTENDED MEMORY BITS
2711	007136	105037	002501		CLRB	†DPB+1	: CLEAR 'MODE' & 'HOR' BITS
2712	007142	105037	002503		CLRB	†DPB+3	: DO 100 ITERATIONS
2713	007146	012737	000144	001162	MOV	†100,†STIMS	: SETUP THE STACK POINTER
2714	007154	012706	001100		TEST4:	MOV	†STACK,SP
2715	007160	004737	025654		JSR	PC,CLRA	: CLEAR THE RP11
2716	007164	005037	002512		CLR	†\$SEC	: CLEAR THE SECTOR - TRACK ADDRESSES
2717	007170	005037	002510		CLR	†\$CYL	: CLEAR THE CYLINDER ADDRESS
2718	007174	012737	043656	002506	MOV	†BUFFER,\$BUF	: SETUP BUFFER ADDRESS
2719	007202	012737	002000	002504	MOV	†1024,†\$WC	: SETUP WORD COUNT
2720	007210	112737	000005	002502	MOVB	†ROSE,†DPB+2	: READ COMMAND
2721	007216	004737	022770		JSR	PC,RP11	: ISSUE THE COMMAND
2722	007222	012737	000001	001220	MOV	†1,†STALL	: LOAD STALL VALUE
2723	007230	004737	023202		JSR	PC,STALL	: STALL FOR 1 MILLISECOND
2724	007234	112764	000005	000004	MOVB	†ROSEK,†RPCS(R4)	: ISSUE READ COMMAND WHILE BUSY
2725	007242	004737	023240		JSR	PC,\$AVR	: STORE THE REGISTERS
2726	007246	032737	002000	001336	BIT	†BIT10,†\$RPER	: DID PROGRAM ERROR SET?
2727	007254	001002			BNE	15	: BRANCH IF SET
2728	007256	104021			ERROR	21	: PROGRAM ERROR DID NOT SET WHEN A
2729							: READ COMMAND WAS ISSUED WHILE
2730							: THE DEVICE WAS BUSY
2731	007260	000412			BR	EXIT4	: BYPASS REST OF TEST
2732	007262	032737	040000	001340	15:	BIT	†ERR,\$RPCS
2733	007270	001001			BNE	25	: 'ERR' SET
2734	007272	104107			ERROR	107	: BR IF NOT
2735	007274	032737	100000	001340	25:	BIT	†ERR,\$RPCS
2736	007302	001001			BNE	EXIT4	: DID 'ERR' SET?
2737	007304	104022			ERROR	22	: BRANCH IF SET
2738	007306	000004			EXIT4:	SCOPE	: 'ERR' DID NOT SET WITH PROGRAM ERROR

†TEST 5 'HEADER NOT FOUND' TEST

UNFORMAT THE FIRST SECTOR ON THE PACK AND THEN READ IT BACK. VERIFY THAT READ AND WRITE HEADER OPERATIONS WILL TRANSFER DATA CORRECTLY. ISSUE A WRITE COMMAND TO SECTOR ZERO; THIS SHOULD CAUSE 'HEADER NOT FOUND' TO SET. REFORMAT THE SECTOR.

†ST5:

2750	007310	033737	001412	001276	BIT	BITS+(\$TN*2),TSTNMS	: IS THIS TEST SELECTED
------	--------	--------	--------	--------	-----	----------------------	-------------------------

2750	007316	001002			BNE	+6	BR IF YES	
2751	007320	000137	010034		JMP	TSTB	GO TO THE NEXT TEST	
2752	007324	012737	007404	001110	MOV	#TESTS, \$LPERR	SETUP THE ERROR LOOP ADDRESS	
2753	007332	012737	007310	001106	MOV	#TESTS, \$LPADR	SETUP THE SCOPE LOOP ADDRESS	
2754	007340	012737	000005	001102	MOV	#S, \$STSTM	SETUP TEST NUMBER AND	
2755							CLEAR THE ERROR FLAG (\$ERFLG)	
2756							LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER	
2757	007346	013777	001102	171566	MOV	\$STSTM, \$DISP	SETUP BYPASS ADDRESS	
2758	007354	012737	010032	001200	MOV	#EXIT5, BYPASS	RP11 IF BUS ADDRESS	
2759	007362	013704	001324		MOV	\$RPAR, \$A	CLEAR ANY EXTENDED MEMORY BITS	
2760	007370	105037	002501		CLRB	DPB+1	CLEAR 'MODE' & 'HOR' BITS	
2761	007378	105037	002503		CLRB	DPB+3	DO 100 ITERATIONS	
2762	007386	012737	000144	001162	MOV	#100, \$TIMES	SETUP THE STACK POINTER	
2763	007404	012706	001100		MOV	\$STACK, \$P	INITIAL LOOP ON ERROR ADDRESS	
2764	007410	012737	007404	001110	MOV	#TESTS, \$LPERR	CLEAR THE RP11	
2765	007416	004737	025654		JSR	PC, CLAP	SETUP INVALID HEADER IMAGE	
2766	007424	012737	000001	043656	MOV	#1, \$BUFFER	CYLINDER/TRACK	
2767	007430	012737	000001	043660	MOV	#1, \$BUFFER+2	SECTOR	
2768	007436	012737	000001	043662	MOV	#1, \$BUFFER+4	LOAD WORD COUNT	
2769	007444	012737	000003	002504	MOV	\$J, \$WC	CLEAR SECTOR/TRACK ADDRESS	
2770	007452	005037	002512		CLR	\$SEC	CLEAR THE CYLINDER ADDRESS	
2771	007458	005037	002510		CLR	\$CYL	'MODE' AND 'HEADER' BITS	
2772	007466	012737	014000	002502	MOV	#MODE, \$HOR, DPB+2	WRITE ORDER	
2773	007470	112737	000003	002502	MOVB	#WATSEK, DPB+2	MISFORMAT SECTOR 0	
2774	007476	004737	022770		JSR	PC, RP11	WAIT FOR THE OPERATION TO COMPLETE	
2775	007502	004737	023664		JSR	PC, CONRDY	STORE THE REGISTERS	
2776	007508	004737	023240		JSR	PC, SAVRP	DID AN ERROR OCCUR?	
2777	007512	032737	100000	001340	BIT	#ERR, \$RPCS	BR IF NOT	
2778	007520	001402			BEG	\$	ERROR OCCURRED ATTEMPTING TO MISFORMAT SECTOR C	
2779	007522	104023			ERROR	\$J	BYPASS TEST & REFORMAT SECTOR C	
2780	007524	000506			BR	TSTB	CHANGE LOOP ON ERROR ADDRESS	
2781	007526	012737	007526	001110	18:	MOV	#IS, \$LPERR	CLEAR THE RP11
2782	007534	004737	025654		JSR	PC, CLAP	CHANGE COMMAND	
2783	007540	112737	000005	002502	MOVB	#ROSEK, DPB+2	EXECUTE THE COMMAND	
2784	007546	004737	022770		JSR	PC, RP11	WAIT FOR CONTROLLER READY	
2785	007552	004737	023664		JSR	PC, CONRDY	ANY ERRORS?	
2786	007558	032737	100000	001340	BIT	#ERR, \$RPCS	BRANCH IF NOT	
2787	007564	001401			BEG	\$	ERROR WHILE READING THE HEADER	
2788	007566	104024			ERROR	\$J	ON SECTOR ZERO	
2789							EXPECTED RESULT	
2790	007570	012737	000001	001124	25:	MOV	#1, \$GDDAT	CLEAR THE INDEX
2791	007576	005000			CLR	\$R	CHECK DATA READ BACK	
2792	007600	023760	001124	043656	35:	CMP	\$GDDAT, \$BUFFER(\$R)	BRANCH ON NON COMPARE
2793	007606	001006			BNE	\$	UPDATE MODIFIER	
2794	007610	062700	000002		ADD	#2, \$R	END OF BUFFER?	
2795	007614	022700	000006		CMP	#6, \$R	BR IF NOT	
2796	007620	001367			BNE	\$	CONTINUE WITH TEST	
2797	007622	000404			BR	TSTSA	GET BAD DATA	
2798	007624	016037	043656	001126	45:	MOV	\$BUFFER(\$R), \$BDDAT	DATA DID NOT VERIFY AFTER READING
2799	007632	104025			ERROR	\$J	THE HEADER OF SECTOR ZERO	
2800							CHANGE LOOP ON ERROR ADDRESS	
2801	007634	012737	007634	001110	TS*5A:	MOV	#TSTSA, \$LPERR	CLEAR THE RP11
2802	007642	004737	025654		JSR	PC, CLAP	CHANGE COMMAND (AND CLEAR 'MODE' & 'HOR'	
2803	007646	012737	000003	002502	MOV	#WATSEK, DPB+2	EXECUTE THE COMMAND	
2804	007654	004737	022770		JSR	PC, RP11	WAIT FOR CONTROLLER READY	
2805	007660	004737	023664		JSR	PC, CONRDY	STORE THE REGISTERS	
2806	007664	004737	023240		JSR	PC, SAVRP		

```

2808 007670 032737 010000 001334 18: BIT #BIT12,$RPLS :DID HEADER NOT FOUND SET?
2809 007676 001002 BNE +6 :BRANCH IF YES
2810 007700 104026 BR ERROR :HNF DID NOT SET
2811 007702 000417 TST58 :BYPASS REST OF TEST
2812 007704 032737 000001 001336 28: BIT #BIT00,$RPER :DID DISK ERROR SET?
2813 007712 001001 BNE +6 :BRANCH IF YES
2814 007714 104027 BR ERROR :DISK DID NOT SET WITH 'HNF'
2815 007716 032737 040000 001340 38: BIT #HE,$RPCS :DID HARD ERROR SET?
2816 007724 001001 BNE +6 :BRANCH IF YES
2817 007726 104030 BR ERROR :HARD ERROR NOT SET AFTER 'HNF'
2818 007730 032737 100000 001340 48: BIT #ERR,$RPCS :DID 'ERR' SET?
2819 007736 001001 BNE +6 :BRANCH IF YES
2820 007740 104031 BR ERROR :'ERR' DID NOT SET WITH 'HNF'
2821 007742 012737 007742 001110 TST58: MOV #TST58,$LPERR :CHANGE LOOP ON ERROR ADDRESS
2822 007750 004737 025654 JSR PC,CLRP :CLEAR THE RP11
2823 007754 005037 043656 CLR BUFFER :SETUP HEADER IMAGE
2824 007760 005037 043660 CLR BUFFER+2 :CYLINDER/TRACK = 0
2825 007764 005037 043662 CLR BUFFER+4 :SECTOR = 0
2826 007770 012737 014000 002502 MOV #MODE!HDR,DPB+2 :'MODE' AND 'HDR' BITS
2827 007776 112737 000003 002502 MOVB #ATSEK,DPB+2 :WRITE COMMAND
2828 010004 004737 022770 JSR PC,RP11 :RESTORE THE HEADER
2829 010010 004737 023664 JSR PC,CONRDY :WAIT FOR THE OPERATION TO COMPLETE
2830 010014 004737 023240 JSR PC,SAVRP :STORE THE REGISTERS
2831 010020 032737 100000 001340 BIT #ERR,$RPCS :DID AN ERROR OCCUR?
2832 010026 001557 BFC EXIT6 :BR IF NOT
2833 010030 104032 BR ERROR :ERROR ATTEMPTING TO RESTORE HEADER
2834 010032 000004
EXIT6: SCOPE

```

```

*****
: *TEST 6 TEST COMMAND ISSUED TO A SEEKING DRIVE
*****
: *ISSUE A SEEK COMMAND AND WAIT FOR DONE TO SET. THEN ISSUE A REAC
: *COMMAND WHILE THE HEADS ARE STILL MOVING. THE RP11E SHOULD
: *HOLD THE WRITE COMMAND TILL THE SEEK IS COMPLETE.

```

```

*****
†TST6:
2845 010034
2846 010034 033737 001414 001276 BIT BITS+($TN*2),TSTNMS :IS THIS TEST SELECTED
2847 010042 001002 BNE +6 :BR IF YES
2848 010044 000137 010370 JMP TST7 :GO TO THE NEXT TEST
2849 010050 012737 010130 001110 MOV #TST6,$LPERR :SETUP THE ERROR LOOP ADDRESS
2850 010056 012737 010034 001106 MOV #TST6,$LPADR :SETUP THE SCOPE LOOP ADDRESS
2851 010064 012737 000006 001102 MOV #6,$TSTNM :SETUP TEST NUMBER AND
2852 :CLEAR THE ERROR FLAG ($ERFLG)
2853 010072 013777 001102 171042 MOV $TSTNM,$DISPLAY :LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
2854 010100 012737 010366 001200 MOV #EXIT6,BYPASS :SETUP BYPASS ADDRESS
2855 010106 013704 001324 MOV RPADR,$R4 :RP11E BUS ADDRESS
2856 010112 105037 002501 CLRB DPB+1 :CLEAR ANY EXTENDED MEMORY BITS
2857 010116 105037 002503 CLRB DPB+3 :CLEAR 'MODE' & 'HDR' BITS
2858 010122 012737 000031 001162 MOV #25,$TIMES :DO 25 ITERATIONS
2859 010130 012706 001100 TEST6: MOV #STACK,SP :SETUP THE STACK POINTER
2860 010134 004737 025654 JSR PC,CLRP :CLEAR THE RP11
2861 010140 005037 002512 CLR $SEC :SECTOR/TRACK 0
2862 010144 005037 002510 CLR $CYL :CYLINDER 0
2863 010150 112737 000015 002502 MCVB #HOMSEK,DPB+2 :HOME SEEK COMMAND

```


F05

```

2864 010156 004737 022770 JSR PC,RP11 ;DO THE COMMAND
2865 010156 004737 023546 JSR PC,DRVDY ;WAIT FOR DRIVE READY
2866 010156 012737 000300 MOV #192,SCYL ;CHANGE CYLINDER TO CYLINDER 192
2867 010174 112737 000011 MOV #SEEK,DPB+2 ;SEEK COMMAND
2868 010202 004737 022770 JSR PC,RP11 ;INITIATE THE SEEK
2869 010206 012737 000001 MOV #1,STALL ;LOAD STALL VALUE
2870 010214 004737 023202 JSR PC,STALL ;STALL FOR 1 MILLISECOND
2871 010220 012764 177775 MOV #-3,RPWC(R4) ;SET WORD COUNT TO THREE WORDS
2872 010226 012764 043656 MOV #BUFFER,RPBA(R4) ;BUFFER ADDRESS
2873 010234 052737 014000 BIS #MODE,HOR,DPB+2 ;SETUP TO READ THE HEADER
2874 010242 112764 000017 MOV #READ,RPCS(R4) ;TRY TO DO A READ
2875 010250 004737 023546 JSR PC,DRVDY ;WAIT FOR DRIVE READY
2876 010254 004737 023240 JSR PC,SAVRP ;STORE THE REGISTERS
2877 010260 032737 000200 BIT #RDY,SRPCS ;DID CONTROLLER BECOME READY ALSO ?
2878 010266 001402 BEQ #16 ;BR IF NOT
2879 010270 104033 ERROR #33 ;READ DIDN'T START DURING SEEK
2880 010272 000436 BR ;EXIT
2891 010274 004737 023664 JSR PC,CONRDY ;WAIT FOR CONTROLLER READY
2892 010300 004737 023240 JSR PC,SAVRP ;STORE THE REGISTERS
2893 010304 032737 100000 BIT #ERR,SRPCS ;ANY ERRORS?
2894 010312 001401 BEQ #25 ;BRANCH IF NO
2895 010314 104034 ERROR #34 ;ERROR OCCURED DURING A READ ISSUED
2896 ; WHILE THE DRIVE IS SEEKING
2897 ; EXPECTED CYLINDER VALUE
2898 ; RECEIVED DATA
2899 ; CORRECT ?
2899 010316 012737 014000 MOV #<192.*32>,SGDDAT ;EXPECTED CYLINDER VALUE
2899 010324 013737 043660 MOV BUFFER+2,$BDDAT ;RECEIVED DATA
2899 010332 023737 001124 CMP $GDDAT,$BDDAT ;CORRECT ?
2899 010340 001366 BNE #25 ;BR IF NOT
2899 010342 005037 CLR $GDDAT ;EXPECTED VALUE FOR SECTOR
2899 010346 013737 043662 MOV BUFFER+4,$BDDAT ;RECEIVED SECTOR
2899 010354 023737 001124 CMP $GDDAT,$BDDAT ;CORRECT ?
2899 010362 001401 BEQ #35 ;BR IF OK
2899 010364 104035 ERROR #35 ;HEADER DATA INCORRECT
2899 010366 000004 EXIT6: SCOPE
2899
2899 *****
2899 ;*TEST 7 TEST 'NXME' BIT
2899 *****
2899 ;*THE NON-EXISTENT MEMORY ERROR BIT IS TESTED BY ATTEMPTING TO READ ;
2899 ;*WORD INTO LOCATION 760000. 'NXME', 'HE', & 'ERR' SHOULD ALL BE SET.
2899 *****
2899 ;*ST7:
2899 BIT BITS+(<STN*2>),TSTNMS ;IS THIS TEST SELECTED
2899 BNE #+6 ;BR IF YES
2899 JMP #TST10 ;GO TO THE NEXT TEST
2899 MOV #TEST7,$LPERR ;SETUP THE ERROR LOOP ADDRESS
2899 MOV #TST7,$LPAOR ;SETUP THE SCOPE LOOP ADDRESS
2899 MOV #7,$TSTNM ;SETUP TEST NUMBER AND
2899 ;CLEAR THE ERROR FLAG ($ERFLG)
2899 MOV $TSTNM,$DISPLAY ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
2899 MOV #EXIT7,BYPASS ;SETUP BYPASS ADDRESS
2899 MOV RPAOR,R4 ;RP11E BUS ADDRESS
2899 CLRB DPB+1 ;CLEAR ANY EXTENDED MEMORY BITS
2899 CLRB DPB+3 ;CLEAR 'MODE' & 'HOR' BITS
2899 MOV #100,$TIMES ;DO 100. ITERATIONS

```



```

010464 012706 001100 TEST7: MOV      $STACK, SP      ; SETUP THE STACK POINTER
010470 004737 025654 JSR      PC, CLR           ; CLEAR THE RP11
010477 012737 000001 002504 MOV      $1, $WC          ; WORD COUNT
010480 005037 002512 CLR      $SEC            ; SECTOR/TRACK ZERO
010486 005037 002510 CLR      $CYL           ; CYLINDER ZERO
010490 112737 000060 002501 MOVVB   $MEX1, MEX0, DPB+1 ; LOAD BOTH EXTENDED MEMORY BITS
010496 012737 160000 002506 MOV      $160000, $BUF    ; LOAD ADDR TO FORCE NON EX MEMOP
010500 112737 000005 002502 MOVVB   $ROSEK, DPB+2    ; TRY TO READ INTO THE LOCATION
010506 004737 022770 JSR      PC, RP11        ; INITIATE THE OPERATION
010510 004737 023664 JSR      PC, CONRDY      ; WAIT FOR CONTROLLER READY
010516 004737 023240 JSR      PC, SAVRP      ; STORE THE REGISTERS
010520 032737 000004 001336 BIT      $BIT02, $RPER   ; DID NON EX MEMORY SET ?
010526 001002 BNE     $ERR           ; BRANCH IF SET
010530 104036 ERROR   $36          ; 'NXME' ERROR DIDN'T SET
010536 000412 BR      EXIT7          ; DON'T CHECK THE RPCS BITS
010540 032737 040000 001340 25: BIT      $HE, $RPCS      ; DID 'HE' SET ON NON EX MEMORY
010546 001001 BNE     $ERR           ; BRANCH IF SET
010550 104037 ERROR   $37          ; 'HE' DIDN'T SET
010556 032737 100000 661340 35: BIT      $ERR, $RPCS     ; DID 'ERR' SET WITH NON EX MEMORY
010560 001001 BNE     EXIT7         ; BRANCH IF SET
010566 104040 ERROR   $40          ; 'ERR' DIDN'T SET
010610 000004 EXIT7:  SCOPE

```

 *TEST 10 SECTOR ADDRESSING TEST

*THE SECTOR ADDRESSING TEST TESTS THE CAPABILITY OF THE RP11E TO LOCATE
 *A SECTOR BY USING THE 'SOT' COUNTER (FOR 'HDR' OPERATIONS) AND TO LOCATE
 *A SECTOR BY COMPARING THE HEADER CONTENTS AGAINST THE CYLINDER AND TRACK,
 *SECTOR ADDRESS REGISTERS (NORMAL MODE). THE TEST IS PERFORMED IN 3 PARTS.

1. WRITE ALL HEADERS ON TRACK 0, CYLINDER 0 IN ASCENDING SEQUENCE FROM INDEX (0, 1, 2 ... 9).
2. READ EACH HEADER, BEGINNING WITH SECTOR 0, AND VERIFY THAT THE SECTOR FIELD IN THE HEADER IS CORRECT FOR THE HEADER EXPECTED.

 VERIFY THAT THE 'SOT' COUNTER AND THE SECTOR ADDRESS IN 'RPDA' ARE EQUAL AND ARE THE EXPECTED VALUE. THE EXPECTED VALUE IS 1 GREATER THAN THE SECTOR READ. THE SECTOR ADDRESS REGISTER AND THE 'SOT' WILL BE INCREMENTED BY THE TIME THE PROGRAM IS ABLE TO READ THE REGISTER.

 WRITE THE SECTOR'S ADDRESS IN THE DATA FIELD. (FOR SECTOR 0, AN OCTAL 12 IS WRITTEN.)

 CONTINUE THIS SEQUENCE FOR THE REMAINING SECTORS ON THE TRACK.
3. READ THE DATA FROM EACH SECTOR USING A NORMAL (I.E. NON-HEADER) READ AND VERIFY THAT THE DATA IS CORRECT FOR THE EXPECTED SECTOR.

```

010612 033737 001420 00127E *TEST10: BIT      BITS+($TN*2), $TSTNMS ; IS THIS TEST SELECTED
010618 001002 BNE     .+6 ; BR IF YES
010620

```

2976	010622	000137	011532		JMP	TST11	GO TO THE NEXT TEST
2977	010626	012737	010706	001110	MOV	#TEST10,\$LPERR	SETUP THE ERROR LOOP ADDRESS
2978	010634	012737	010612	001106	MOV	#TEST10,\$LPAOR	SETUP THE SCOPE LOOP ADDRESS
2979	010642	012737	000010	001102	MOV	#10,\$STNM	SETUP TEST NUMBER AND
2980							CLEAR THE ERROR FLAG (\$ERFLG)
2981	010650	013777	001102	170264	MOV	\$STNM,\$DISPLAY	LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
2982	010656	012737	011530	001200	MOV	#EXIT10,\$BYPASS	SETUP BYPASS ADDRESS
2983	010664	013704	001324		MOV	RPADR,R4	RP11E BUS ADDRESS
2984	010670	105037	002501		CLRB	DPB+1	CLEAR ANY EXTENDED MEMORY BITS
2985	010674	105037	002503		CLRB	DPB+3	CLEAR 'MODE' & 'HDR' BITS
2986	010700	012737	000031	001162	MOV	#25,\$TIMES	DO 25 ITERATIONS
2987	010706	012706	001100		MOV	#STACK,\$SP	RESET THE STACK POINTER
2988	010712	012737	010706	001110	MOV	#TEST10,\$LPERR	LOOP ON ERROR ADDRESS
2989	010720	004737	025654		JSR	PC,\$CLAP	CLEAR THE RP11
2990	010724	012737	043656	002506	MOV	#BUFFER,\$BUF	SETUP BUFFER ADDRESS
2991	010732	012737	000036	002504	MOV	#30,\$WC	SETUP WORD COUNT FOR 10 SECTORS
2992	010740	005037	002510		CLR	\$CYL	SETUP FOR CYLINDER 0
2993	010744	005037	002512		CLR	\$SEC	TRACK/SECTOR 0
2994	010750	004737	024304		JSR	PC,\$LDDSEC	SETUP FORMAT BUFFER
2995	010754	012737	014000	002502	MOV	#MODE!HDR,DPB+2	SETUP THE HEADER CONTROL BITS
2996	010762	112737	000003	002502	MOV	#WRTSEK,DPB+2	THE COMMAND CODE
2997	010770	004737	022770		JSR	PC,\$RP11	FORMAT THE TRACK
2998	010774	004737	023664		JSR	PC,\$CONRDY	WAIT FOR THE ORDER TO COMPLETE
2999	011000	004737	023240		JSR	PC,\$SAVRP	STORE THE REGISTERS
3000	011004	032737	100000	001340	BIT	#ERR,\$RPCS	ERROR ?
3001	011012	001403			BEG	15	BR IF NOT
3002	011014	104041			ERROR	41	ERROR TRYING TO FORMAT THE TRACK
3003	011016	000137	011530		JMP	EXIT10	BYPASS REST OF THE TEST
3004	011022	105037	002512		CLRB	\$SEC	START CHECK WITH SECTOR 0
3005	011026	012737	011026	001110	MOV	#25,\$LPERR	CHANGE THE LOOP ON ERROR ADDRESS
3006	011034	004737	025654		JSR	PC,\$CLAP	CLEAR THE RP11
3007	011040	012737	014000	002502	MOV	#MODE!HDR,DPB+2	SET 'MODE' AND 'HDR' BITS
3008	011046	112737	000005	002502	MOV	#RDSEK,DPB+2	CHANGE THE OPERATION TO A READ
3009	011054	012737	000003	002504	MOV	#3,\$WC	CHANGE THE WORD COUNT TO 3
3010	011062	004737	022770		JSR	PC,\$RP11	READ THE SECTOR'S HEADER
3011	011066	004737	023664		JSR	PC,\$CONRDY	WAIT FOR THE ORDER TO COMPLETE
3012	011072	004737	023240		JSR	PC,\$SAVRP	STORE THE REGISTERS
3013	011076	012737	000000	001220	MOV	#0,\$STALLT	LOAD STALL VALUE
3014	011104	004737	023202		JSR	PC,\$STALL	STALL FOR 0 MILLISECONDS
3015	011110	016437	000014	001350	MOV	RPDA(R4),\$RPDA	GET THE CONTENTS OF THE ADDRESS REGISTER
3016	011116	032737	100000	001340	BIT	#ERR,\$RPCS	ERROR ?
3017	011124	001402			BEG	35	BR IF NOT
3018	011126	104042			ERROR	42	ERROR READING THE HEADER
3019	011130	000457			BR	65	BYPASS DATA CHECK
3020	011132	005037	001124		CLR	\$GDDAT	CLEAR FOR EXPECTED DATA
3021	011136	113737	002512	001124	MOV	\$SEC,\$GDDAT	MOVE SECTOR ADDRESS
3022	011144	023737	001124	043662	CMP	\$GDDAT,\$BUFFER+4	CHECK THE SECTOR FIELD
3023	011152	001401			BEG	45	BR IF CORRECT
3024	011154	104043			ERROR	43	SECTOR FIELD FROM HEADER NOT CORRECT
3025	011156	005237	001124		INC	\$GDDAT	INCREMENT SECTOR ADDRESS FOR 'SOT' CHECK
3026	011162	023727	001124	000012	CMP	\$GDDAT,#10	EXCEED THE MAXIMUM ?
3027	011170	103402			BLO	+6	BR IF NOT
3028	011172	005037	001124		CLR	\$GDDAT	RESET COMPARISON VALUE
3029	011176	013737	001350	001126	MOV	\$RPDA,\$BDDAT	CHECK THE 'SOT'
3030	011204	042737	177417	001126	BIC	#1C360,\$BDDAT	CLEAR THE OTHER BITS
3031	011212	006237	001126		ASR	\$BDDAT	SHIFT THE 'SCT'

3032	011216	006237	001126			ASR	\$BDDAT	:SHIFT THE 'SOT'
3033	011222	006237	001126			ASR	\$BDDAT	:SHIFT THE 'SOT'
3034	011226	006237	001126			ASR	\$BDDAT	:SHIFT THE 'SOT'
3035	011232	013737	001350	001122		MOV	\$APDA,\$BDAADR	:CHECK THE SECTOR ADDRESS
3036	011240	042737	177760	001122		BIC	#17,\$BDAADR	:LEAVE THE SECTOR ADDRESS
3037	011246	023737	001124	001126		CMP	\$GDDAT,\$BDDAT	:IS THE 'SOT' CORRECT ?
3038	011254	001004				BNE	5\$:BR IF NOT
3039	011256	023737	001126	001122		CMP	\$BDDAT,\$BDAADR	:DOES THE SECTOR ADDRESS EQUAT THE 'SOT' ?
3040	011264	001401				BEQ	6\$:BR IF IT DOES
3041	011266	104045			5\$:	ERROR	45	: 'SOT' OR SECTOR ADDRESS IS NOT CORRECT
3042	011270	012737	011270	001110	6\$:	MOV	#6\$,\$LPERR	:CHANGE THE LOOP ON ERROR ADDRESS
3043	011276	004737	025654			JSR	PC,CLRP	:CLEAR THE RP11
3044	011302	105037	002503			CLRB	DPB+3	:CLEAR 'MODE' & 'HDR' BITS
3045	011306	004737	024336			JSR	PC,FILSEC	:SETUP TO WRITE SECTOR ADDRESS AS DATA
3046	011312	112737	000003	002502		MOVB	#WATSEK,DPB+2	:CHANGE OPERATION
3047	011320	004737	022770			JSR	PC,RP11	:WRITE THE SECTOR
3048	011324	004737	023664			JSR	PC,CONRDY	:WAIT FOR THE ORDER TO COMPLETE
3049	011330	004737	023240			JSR	PC,SAVRP	:STORE THE REGISTERS
3050	011334	032737	100000	001340		BIT	#ERR,\$RPCS	:ERROR ?
3051	011342	001401				BEQ	7\$:BR IF NOT
3052	011344	104044				ERROR	44	:ERROR TRYING TO WRITE SECTOR ADDRESS AS DATA
3053	011346	112737	000005	002502	7\$:	MOVB	#RDSEK,DPB+2	:CHANGE THE OPERATION
3054	011354	105237	002512			INCB	\$SEC	:INCREMENT THE SECTOR
3055	011360	123727	002512	000012		CMPB	\$SEC,#10.	:AT MAXIMUM SECTOR ADDRESS
3056	011366	001217				BNE	2\$:BR IF NOT
3057	011370	012737	011424	001110		MOV	#8\$,\$LPERR	:CHANGE THE LOOP ON ERROR ADDRESS
3058	011376	012737	000400	002504		MOV	#256,\$WC	:CHANGE WORD COUNT
3059	011404	105037	002512			CLRB	\$SEC	:START WITH SECTOR ZERO
3060	011410	012737	000012	001124		MOV	#10,\$GDDAT	:SECTOR ZERO PATTERN
3061	011416	112737	000005	002502		MOVB	#RDSEK,DPB+2	:CHANGE OPERATION CODE
3062	011424	004737	025654		8\$:	JSR	PC,CLRP	:CLEAR THE RP11
3063	011430	004737	022770			JSR	PC,RP11	:READ THE SECTOR
3064	011434	004737	023664			JSR	PC,CONRDY	:WAIT FOR THE ORDER TO COMPLETE
3065	011440	004737	023240			JSR	PC,SAVRP	:STORE THE REGISTERS
3066	011444	032737	100000	001340		BIT	#ERR,\$RPCS	:ERROR ?
3067	011452	001402				BEQ	9\$:BR IF NOT
3068	011454	104017				ERROR	17	:ERROR TRYING TO READ THE SECTOR
3069	011456	000412				BR	10\$:BYPASS DATA CHECK
3070	011460	005037	001120		9\$:	CLR	\$GDADR	:CLEAR FOR EXPECTED DATA
3071	011464	113737	002512	001120		MOVB	\$SEC,\$GDADR	:EXPECTED DATA
3072	011472	023737	001124	043656		CMP	\$GDDAT,BUFFER	:CORRECT CONTENTS ?
3073	011500	001401				BEQ	10\$:BR IF CORRECT
3074	011502	104046				ERROR	46	:SECTOR PATTERN NOT CORRECT FOR SECTOR ADDRESS
3075	011504	105237	002512		10\$:	INCB	\$SEC	:INCREMENT SECTOR ADDRESS
3076	011510	123727	002512	000012		CMPB	\$SEC,#10.	:FINISHED SECTOR NINE ?
3077	011516	001404				BEQ	EXIT10	:BR IF COMPLETED
3078	011520	113737	002512	001124		MOVB	\$SEC,\$GDDAT	:NEXT SECTOR
3079	011526	000736				BR	8\$:CONTINUE CHECKING
3080	011530	000004				EXIT10:	SCOPE	:LOOP ?

: *TEST 11 TRACK ADDRESSING TEST

: *TRACK ADDRESSING IS TESTED BY WRITING THE TRACK'S ADDRESS AS DATA IN
: *SECTOR 0 OF EACH TRACK IN CYLINDER 0. SECTOR 0 IS THEN READ BACK AND

3081
3082
3083
3084
3085
3086
3087

3174
3175
3176
3177
3178
3179
3180
3181
3182
3183
3184
3185
3186
3187
3188
3189
3190
3191
3192
3193
3194
3195
3196
3197
3198
3199

: TEST 12 EXTENDED MEMORY ADDRESS TEST

: *THIS TEST TESTS THE OPERATION OF THE EXTENDED MEMORY ADDRESS BITS. IF THE
: *SYSTEM DOES NOT HAVE MEMORY MANAGEMENT OR HAS MEMORY MANAGEMENT AND ONLY
: *32K, THE TEST WILL NOT BE PERFORMED.

1. THE PROGRAM WRITES A 2 WORD TEST SECTOR OF ALL ONES.
2. EXTENDED ADDRESS BIT 'MEX00' IS TESTED BY CLEARING LOCATION 200000 AND READING THE TEST SECTOR INTO LOCATION 177776. LOCATION 200000 IS CHECKED TO VERIFY THAT THE DATA IS CORRECT (ONES). THE PROGRAM VERIFIES THAT 'MEX00' HAS SET AND THAT 'MEX01' IS NOT SET.
3. IF THE SYSTEM HAS AT LEAST 64K, 'MEX01' IS TESTED. LOCATION 400000 IS CLEARED AND THE TEST SECTOR IS READ INTO LOCATION 377776 ('MEX00' IS SET FOR THE READ). LOCATION 400000 IS CHECKED FOR THE PROPER CONTENTS (ONES). 'MEX00' SHOULD HAVE RESET AND 'MEX01' SHOULD BE SET.

†ST12:

012062			
012062	033737	001424	001276
012070	001002		
012072	000137	012720	
012076	012737	012202	001110
012104	012737	012062	001106
012112	012737	000012	001102
012120	013777	001102	167014
012126	012737	012712	001200
012134	013704	001324	
012140	105037	002501	
012144	105037	002503	
012150	012737	000144	001162
012156	005737	022232	
012162	100402		
012164	000137	012720	
012170	005737	001232	
012174	001002		
012176	000137	012720	
012202	012706	001100	
012206	012737	012202	001110
012214	004737	025654	
012220	005037	043656	
012224	012737	177777	043660
012232	012737	043656	002506
012240	012737	000002	002504
012246	005037	002512	
012252	005037	002510	
012256	012737	000003	002502
012264	004737	022770	
012270	004737	023664	
012274	004737	023240	

```

BIT          BITS+(<STN*2>),TSTNMS  IS THIS TEST SELECTED
BNE          +6                       BR IF YES
JMP          TST13                     GO TO THE NEXT TEST
MOV          #TEST12,$LPERR            SETUP THE ERROR LOOP ADDRESS
MOV          #TST12,$LPADR            SETUP THE SCOPE LOOP ADDRESS
MOV          #12,$STNM                SETUP TEST NUMBER AND
                                       CLEAR THE ERROR FLAG ($ERFLG)
MOV          $STNM,$DISPLAY           LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
MOV          #EXIT12,BYPASS           SETUP BYPASS ADDRESS
MOV          RPADR,R4                 RP11E BUS ADDRESS
CLRB        DPB+1                     CLEAR ANY EXTENDED MEMORY BITS
CLRB        DPB+3                     CLEAR 'MODE' & 'HDR' BITS
MOV          #100,$TIMES              DO 100 ITERATIONS
TST         $KT11                     MEMORY MANAGEMENT ON THE SYSTEM ?
BMI         IS                         BR IF ON SYSTEM
JMP         TST13                     GO TO THE NEXT TEST
IS:         TST         MAXMEM+2        ENOUGH MEMORY FOR THIS TEST ?
BNE         TEST12                    BR IF YES
JMP         TST13                     GO TO THE NEXT TEST
TEST12:    MOV          #STACK,SP      SETUP THE STACK POINTER
MOV          #TEST12,$LPERR           INITIAL ERROR LOOP ADDRESS
JSR         PC,CLRP                   CLEAR THE RP11
CLR        BUFFER                     TEST PATTERN
MOV        #-1,BUFFER+2              SECOND WORD OF PATTERN
MOV        #BUFFER,$BUF              SETUP THE BUFFER ADDRESS
MOV        #2,$WC                     WORD COUNT OF 2
CLR        $SEC                       SECTOR/TRACK ZERO
CLR        $CYL                       CYLINDER ZERO
MOV        #WRTSEK,DPB+2              WRITE COMMAND
JSR        PC,RP11                    INITIATE THE COMMAND
JSR        PC,CONRDY                  WAIT FOR CONTROLLER READ
JSR        PC,SAVRP                   STORE THE REGISTERS

```

3200	012300	032737	100000	001340		BIT	#ERR, \$RPCS	: DID OPERATION TERMINATE WITH AN ERROR ?
3201	012306	001403				BEG	15	: BR IF NOT
3202	012310	104004				ERROR	4	: ERROR AFTER 2 WORD WRITE
3203	012312	000137	012712			JMP	EXIT12	: BYPASS REST OF TEST
3204	012316	012737	012316	001110	15:	MOV	#15, \$LPERR	: CHANGE LOOP ON ERROR ADDRESS
3205	012324	004737	025654			JSR	PC, CLRP	: CLEAR THE RP11
3206	012330	012737	177600	172356		MOV	#177600, KIPAR7	: OPEN I/O REGISTERS
3207	012336	005037	172340			CLR	KIPAR0	: ENABLE FIRST 4K
3208	012342	012737	000200	172342		MOV	#200, KIPAR1	: ENABLE SECOND 4K
3209	012350	012737	002000	172344		MOV	#2000, KIPAR2	: ENABLE FIRST TEST LOCATION
3210	012356	012737	077406	172300		MOV	#77406, KIPDR0	: LOAD DESCRIPTOR REGISTER 0
3211	012364	012737	077406	172302		MOV	#77406, KIPDR1	: LOAD DESCRIPTOR REGISTER 1
3212	012372	012737	077406	172304		MOV	#77406, KIPDR2	: LOAD DESCRIPTOR REGISTER 2
3213	012400	012737	077406	172316		MOV	#77406, KIPDR7	: LOAD DESCRIPTOR REGISTER 7
3214	012406	012737	000001	177572		MOV	#1, \$SRA0	: TURN ON MEMORY MANAGEMENT
3215	012414	005037	040000			CLR	#40000	: CLEAR LOCATION 200000
3216	012420	012737	177776	002506		MOV	#177776, \$BUF	: SETUP BUS ADDR
3217	012426	112737	000005	002502		MOVB	#RDSEK, DPB+2	: CHANGE COMMAND TO READ
3218	012434	004737	022770			JSR	PC, RP11	: START THE COMMAND
3219	012440	004737	023664			JSR	PC, CONRDY	: WAIT FOR CONTROLLER READY
3220	012444	004737	023240			JSR	PC, SAVRP	: STORE THE REGISTERS
3221	012450	032737	100000	001340		BIT	#ERR, \$RPCS	: ANY ERRORS?
3222	012456	001401				BEG	25	: BRANCH IF NOT
3223	012460	104050				ERROR	50	: ERROR AFTER READING 2 WORDS
3224	012462	032737	000020	001340	25:	BIT	#MEX0, \$RPCS	: DID 'MEX0' SET ?
3225	012470	001001				BNE	35	: BRANCH IF IT DID
3226	012472	104051				ERROR	51	: 'MEX0' DID NOT SET AFTER 2 WORD READ
3227								: STARTING AT LOCATION 177776
3228	012474	032737	000040	001340	35:	BIT	#MEX1, \$RPCS	: 'MEX1' SHOULD NOT BE SET
3229	012502	001401				BEG	45	: BRANCH IF NOT SET
3230	012504	104052				ERROR	52	: 'MEX1' IS SET - SHOULD NOT BE
3231	012506	022737	177777	040000	45:	CMP	#177777, #40000	: WAS DATA READ INTO LOCATION
3232	012514	001407				BEG	TST128	: 200000 CORRECTLY? - BRANCH IF YES
3233	012516	012737	177777	001124		MOV	#177777, \$GDDAT	: EXPECTED CONTENTS
3234	012524	013737	040000	001126		MOV	#40000, \$BDDAT	: CONTENTS OF LOCATION 200000
3235	012532	104053				ERROR	53	: DATA COMPARE ERROR AT LOC 200000
3236	012534	023727	001232	000002	TST128:	CMP	MAXMEM+2, #2	: ENOUGH MEMORY FOR THE NEXT PART OF THE TEST ?
3237	012542	103463				BLO	EXIT12	: BR IF NOT
3238	012544	012737	004000	172344		MOV	#4000, KIPAR2	: CHANGE TEST LOCATION
3239	012552	012737	012552	001110	15:	MOV	#15, \$LPERR	: CHANGE THE LOOP ON ERROR ADDRESS
3240	012560	004737	025654			JSR	PC, CLRP	: CLEAR THE RP11
3241	012564	005037	040000			CLR	#40000	: CLEAR LOCATION 400000
3242	012570	012737	177776	002506		MOV	#177776, \$BUF	: BUFFER ADDRESS
3243	012576	112737	000020	002501		MOVB	#MEX0, DPB+1	: SETUP EXTENDED MEMORY ADDRESS BIT
3244	012604	112737	000005	002502		MOVB	#RDSEK, DPB+2	: READ COMMAND
3245	012612	004737	022770			JSR	PC, RP11	: START THE COMMAND
3246	012616	004737	023664			JSR	PC, CONRDY	: WAIT FOR CONTROLLER READY
3247	012622	004737	023240			JSR	PC, SAVRP	: STORE THE REGISTERS
3248	012626	032737	100000	001340		BIT	#ERR, \$RPCS	: ANY ERRORS?
3249	012634	001401				BEG	25	: BRANCH IF NOT
3250	012636	104054				ERROR	54	: ERROR WHILE READING TWO WORDS
3251	012640	032737	000020	001340	25:	BIT	#MEX0, \$RPCS	: IS 'MEX0' SET ?
3252	012646	001401				BEG	35	: BRANCH IF NOT
3253	012650	104055				ERROR	55	: 'MEX0' DID NOT CLEAR AFTER 2 WORD
3254								: READ STARTING AT LOCATION 377776
3255	012652	032737	000040	001340	35:	BIT	#MEX1, \$RPCS	: DID 'MEX1' SET

M05

```

3256 012660 001001 BNE 45 ;BR IF SET
3257 012662 104056 ERROR 56 ;'MEXI' DID NOT SET AFTER 2 WORD TRANSFER
3258 ; STARTING AT LOCATION 377776
3259 012664 022737 177777 040000 45: CMP #177777, @#40000 ; WAS DATA READ INTO LOCATION 40000C
3260 012672 001407 BEQ EXIT12 ; CORRECTLY ? - BRANCH IF YES
3261 012674 012737 177777 001124 MOV #177777, $GDDAT ; EXPECTED DATA
3262 012702 013737 040000 001126 MOV @#40000, $BDDAT ; DATA FROM LOCATION 400000
3263 012710 104057 ERROR 57 ; DATA COMPARE ERROR AT LOC 400000
3264 012712 005037 177572 EXIT12: CLR @#SR0 ; TURN OFF MEMORY MANAGEMENT
3265 012716 000004 SCOPE ; LOOP ?

```

```

:*****
: *TEST 13 DATA BUFFER REGISTER BIT TEST

```

```

: *THE DATA BUFFER REGISTER TEST VERIFIES THAT ALL 36 BITS OF THE DATA
: *BUFFER REGISTER CAN BE SET AND CLEARED.

```

```

: * 1. THE TEST FIRST WRITES THE FOLLOWING TEST PATTERN IN CYLINDER 0,
: * TRACK 0, SECTOR 0 USING 10/15 MODE ('MODE' BIT SET).

```

```

: *
: * 052525
: * 052525
: * 000005
: * 125252
: * 125252
: * 000012
: * 052525
: * 052525
: * 000005
: * 125252

```

```

: *
: * .
: *
: * ETC

```

```

: * 2. THE TEST SECTOR IS THEN READ AND THE DATA IS COMPARED. AT THE
: * COMPLETION OF THE TEST, THE PROGRAM RESTORES THE TEST SECTOR TO
: * PDP-11 MODE.

```

```

:*****
: *TST13:

```

```

3297 012720 033737 001426 001276 BIT BITS+(<$TN*2>), TSTNMS ; IS THIS TEST SELECTED
3298 012720 001002 BNE +6 ; BR IF YES
3299 012726 000137 JMP TST14 ; GO TO THE NEXT TEST
3300 012730 012737 013574 001110 MOV #TST13, $LPERR ; SETUP THE ERROR LOOP ADDRESS
3301 012734 012737 012720 001106 MOV #TST13, $LPADR ; SETUP THE SCOPE LOOP ADDRESS
3302 012742 012737 000013 001102 MOV #13, $TSTNM ; SETUP TEST NUMBER AND
3303 ; CLEAR THE ERROR FLAG ($ERFLG)
3304 ; LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
3305 012756 013777 001102 166156 MOV $TSTNM, @DISPLAY
3306 012764 012737 013572 001200 MOV #EXIT13, BYPASS ; SETUP BYPASS ADDRESS
3307 012772 013704 001324 MOV RPADR, R4 ; RP11E BUS ADDRESS
3308 012776 105037 002501 CLRB DPB+1 ; CLEAR ANY EXTENDED MEMORY BITS
3309 013002 105037 002503 CLRB DPB+3 ; CLEAR 'MODE' & 'HDR' BITS
3310 013006 012737 000144 001162 MOV #100, $TIMES ; DO 100. ITERATIONS
3311 013014 012706 001100 TEST13: MOV #STACK, SP ; SETUP THE STACK POINTER

```

3312	013020	012701	043656		MOV	#BUFFER, R1	: BUFFER ADDRESS
3313	013024	012702	000200		MOV	#128, R2	: SIZE
3314	013030	012721	052525	15:	MOV	#52525, (R1)+	: PATTERN FOR BITS 20 - 35
3315	013034	012721	052525		MOV	#52525, (R1)+	: PATTERN FOR BITS 4 - 19
3316	013040	012721	000005		MOV	#5, (R1)+	: PATTERN FOR BITS 0 - 3
3317	013044	005302			DEC	R2	: AT THE END ?
3318	013046	001410			BFO	R2	: BR IF END
3319	013050	012721	125252		MOV	#125252, (R1)+	: PATTERN FOR BITS 20 - 35
3320	013054	012721	125252		MOV	#125252, (R1)+	: PATTERN FOR BITS 4 - 19
3321	013060	012721	000012		MOV	#12, (R1)+	: PATTERN FOR BITS 0 - 3
3322	013064	005302			DEC	R2	: AT THE END ?
3323	013066	001360			BNE	15	: BR IF NOT
3324	013070	012737	013070	001110	MOV	#25, \$LPERR	: LOOP ON ERROR ADDRESS
3325	013076	004737	025654		JSR	PC, CLR	: CLEAR THE RP11


```

013575 032777 000020 165456
013576 001401
013577 005302
013578 001323
013579 000401
013580 000000
013581 012737 001110
013582 004737
013583 005037
013584 012737 002504
013585 005037 002502
013586 005037
013587 012737 002506
013588 004737
013589 004737
013590 004737 001340
013591 001401
013592 004060
013593 000004

```

```

95: BIT BSH04 JSNR ; CONTINUE COMPARING ANYWAY ?
      BPO ST13A ; OR IF NOT
      DEC ST13A ; FINISHED COMPARING ?
      BNE ST13A ; OR IF NOT
      BSR ST13A ; CHANGE MODE OF THE SECTOR
      WORD 0 ; COMPARISON ERROR COUNTER
105: ST13A: MOV @ST13A, SLPERR ; CHANGE LOOP ON ERROR ADDRESS
      JSR PC, CLAP ; CLEAR THE RP11
      CLRB DPB+3 ; CLEAR THE 'MODE' & 'MOP' BITS
      MOV B2, SLC ; CHANGE WORD COUNT
      MOVB @RATSEK, DPB+2 ; CHANGE COMMAND
      CLR BUFFER ; SET DATA TO ZERO
      CLRB BUFFER+2 ; OTHER DATA WORD
      MOV BUFFER, SBUF ; BUFFER ADDRESS
      JSR PC, RP11 ; RE-WRITE THE SECTOR
      JSR PC, CONADY ; WAIT FOR THE OPERATION TO COMPLETE
      JSR PC, SAVRP ; STORE THE REGISTERS
      BIT @RAT, SRPCS ; DID THE WRITE FINISH WITHOUT ERROR ?
      BPO EXIT13 ; OR IF IT DID
      BSR ERROR ; ERROR RE-WRITING THE SECTOR
EXIT13: SCOPE ; LOOP ?

```

*TEST 14 DATA STORAGE/RETRIEVAL TEST

*THE DATA TEST PERFORMS DATA STORAGE AND RETRIEVAL TESTING USING THE CONTROLLER AND THE CURRENTLY SELECTED DRIVE. UNLESS ALTERED BY THE OPERATOR, THE TEST WILL WRITE AND WRITE CHECK THE ENTIRE DISK PACK USING ALL 16 DATA PATTERNS IN A ROTATING MANNER. THE BUFFER SIZE USED FOR THE WRITE AND WRITE CHECK ORDERS IS DETERMINED BY THE AVAILABLE MEMORY ON THE SYSTEM (MINUS THE SPACE REQUIRED BY THE PROGRAM LOADER). THE PROGRAM WILL USE THE AVAILABLE MEMORY SIZE OR 8K, WHICHEVER IS LESS, AS THE BUFFER SIZE.

*AFTER THE PACK HAS BEEN WRITTEN, THE TEST WILL READ AND COMPARE THE DATA. THE BUFFER SIZE USED FOR THE READ IS THE SAME SIZE AS THAT USED FOR THE WRITE/WRITE CHECK PART OF THE TEST. FOR THE READ, THE TEST WILL ROTATE THE READ BUFFER THROUGH MEMORY SO THAT ALL OF THE AVAILABLE MEMORY IS USED BY THE READ AND COMPARE OPERATIONS. THE TEST USES MEMORY MANAGEMENT ON SYSTEMS WITH MORE THAN 28K. (MEMORY MANAGEMENT WILL NOT BE USED IF ON MEMORY MANAGEMENT SYSTEMS WITH 28K OR LESS.)

*THE DATA PATTERNS USED BY THE DATA TEST ARE GIVEN BELOW:

PAT 0	PAT 1	PAT 2	PAT 3	PAT 4	PAT 5	PAT 6	PAT 7
000000	000001	177776	000000	000000	052525	007417	026455
000000	000003	177774	000000	010421	052525	007417	026455
000000	000007	177770	000000	021042	052525	007417	026455
000000	000017	177760	177777	031463	125252	170360	151322
000000	000037	177740	177777	042104	125252	170360	151322
000000	000077	177700	177777	052525	125252	170360	151322
000000	000177	177600	000000	063146	052525	007417	026455
000000	000377	177400	000000	073567	052525	007417	026455
000000	000777	177000	177777	104210	125252	170360	151322
000000	001777	176000	177777	114631	125252	170360	151322

E06

35:	CLR	LOMEM+2	: CLEAR UPPER STORAGE WORD
	MOV	LOMEM,ACTMEM	: STARTING ADDRESS OF FIRST BUFFER
	MOV	LOMEM+2,ACTMEM+2	: UPPER ADDRESS BITS
	MOV	WC14,BLKS14	: CONVERT THE WORD COUNT TO A SECTOR COUNT
	SHAB	BLKS14	: RIGHT JUSTIFY THE SECTOR COUNT
	CLRB	BLKS14+1	: CLEAR THE RESIDUE
	INC	BLKS14	: CHANGE TO ABSOLUTE COUNT
	MOV	BLKS14,-(SP)	: CONVERT BLOCK COUNT TO SECTOR/TRACK INCREMENTS
	INCRK		: CLEAR TRACK STORAGE
	CLR	INCSEC	: CLEAR SECTOR STORAGE
45:	SUB	#10,-(SP)	: SUBTRACT A FULL SECTOR
	BIT	55	: BR IF MINUS
	INC	INCRK	: COUNT THE TRACK
	BR	45	: CONTINUE
55:	ADD	#10,-(SP)	: CORRECT THE SECTOR RESIDUE
	MOV	(SP)+1,INCSEC	: SECTOR INCREMENT VALUE
TEST14:	MOV	#STACK,SP	: LOAD THE STACK POINTER
	JSR	PC,SECNBR	: GET THE NUMBER OF SECTORS ON THE PACK BEING USED
15:	MOV	SSEC,\$SEC	: STARTING SECTOR ADDRESS
	MOV	STRK,\$TRK	: STARTING TRACK ADDRESS
	MOV	SCYL,\$CYL	: STARTING CYLINDER ADDRESS
	MOV	LOMEM,\$BUF	: BUFFER ADDRESS
	MOV	WC14,\$WC	: STARTING WORD COUNT
	MOV	#15,PATNUM	: PRESET THE PATTERN INDEX
	TST	OPRSEL	: USING AN OPERATOR SPECIFIED ADDRESS ?
	BNE	25	: BR IF YES
	SUB	BLKS14,PAKSIZ	: DECREMENT THE SECTOR COUNT FOR THE FIRST OPERATION
	SBC	PAKSIZ+2	: SELECT THE PATTERN
25:	JSR	PC,PATSEL	: LOOP ON ERROR ADDRESS
35:	MOV	#3,\$LERR	: CLEAR THE RP11
	JSR	PC,CLRP	: INHIBIT WRITE AND WRITE CHECK ?
	MOV	JSWR,-(SP)	
	COM	(SP)	
	BIT	#SW00!SW01,(SP)+	: YES--BRANCH
	BEG	TST14	: MOVE DATA PATTERN INTO THE BUFFER
	JSR	PC,SETBUF	: INHIBIT WRITE ?
	BIT	#SW00,\$SWR	: YES--BRANCH
	BNE	45	: COMMAND=WRITE
	MOV	#WRTSEK,DPB+2	: DO THE COMMAND
	JSR	PC,RP11	: WAIT FOR THE COMMAND TO COMPLETE
	JSR	PC,CONRDY	: STORE THE REGISTERS
	JSR	PC,SAVRP	: ERROR OCCURED ?
	BIT	#ERR,\$RPCS	: BR IF NOT
	BEG	45	: REPORT THE ERROR
	ERROR	61	: INHIBIT WRITE CHECK ?
45:	BIT	#SW01,\$SWR	: YES--BRANCH
	BNE	55	: COMMAND=WRITE CHECK
	JSR	PC,CLRP	: DO THE WRITE CHECK
	MOV	#WATCHK,DPB+2	: WAIT FOR THE COMMAND TO FINISH
	JSR	PC,RP11	: STORE THE REGISTERS
	JSR	PC,CONRDY	: ERROR ?
	JSR	PC,SAVRP	: BR IF NOT
	BIT	#ERR,\$RPCS	: REPORT THE WRITE CHECK ERROR
	BEG	55	
	ERROR	5	

F06

DZRPYC.P11

09-SEP-77 16:03

T14

DATA STORAGE/RETRIEVAL TEST

3550	014334	004737	024114	58:	JSR	PC, INCADR	: INCREMENT THE ADDRESS	
3551	014340	000710			BR	28	: CONTINUE	
3552	014342	004737	024002	TST14A:	JSR	PC, SECNBR	: GET NUMBER OF SECTORS ON THE DRIVE	
3553							: BEING TESTED	
3554	014346	113737	001272		MOVB	SSEC, SSEC	: STARTING SECTOR	
3555	014354	113737	001270		MOVB	STRK, STRK	: STARTING TRACK	
3556	014356	013737	001266		MOV	SCYL, SCYL	: STARTING CYLINDER	
3557	014370	013737	001256		MOV	WC14, SWC	: INITIAL WORD COUNT	
3558	014378	012737	000017		MOV	B15, PATNUM	: PRESET PATTERN NUMBER	
3559	014404	005737	001260		TST	OPRSEL	: USING AN OPERATOR SPECIFIED PATTERN	
3560	014410	001005			BNE	18	: BR IF YES	
3561	014412	163737	001254		SUB	BLKS14, PAKSIZ	: DECREMENT THE SECTOR COUNT FOR	
3562	014420	005637	001246		SBC	PAKSIZ+2	: THE FIRST OPERATION	
3563	014424	005037	001314	18:	CLR	RETRY	: CLEAR THE RETRY COUNTER	
3564	014430	004737	024050		JSR	PC, PATSEL	: SELECT A PATTERN	
3565	014434	004737	024502		JSR	PC, GETBUF	: GENERATE A BUFFER ADDRESS	
3566	014440	012737	014440	28:	MOV	B28, \$LPERR	: LOOP ON ERROR ADDRESS	
3567	014446	004737	025654		JSR	PC, CLAP	: CLEAR THE RP11	
3568	014452	032777	000004		BIT	#SW02, JSWR	: INHIBIT READ DATA AND SOFTWARE COMPARE	
3569	014460	001056			BNE	EXIT14	: YES--BRANCH	
3570	014462	112737	000005		MOVB	#ROSEK, DPB+2	: LOAD READ COMMAND	
3571	014470	004737	022770		JSR	PC, RP11	: START THE READ	
3572	014474	004737	023664		JSR	PC, COMRDY	: WAIT FOR THE READ TO COMPLETE	
3573	014500	004737	023240		JSR	PC, SAVRP	: STORE THE REGISTERS	
3574	014504	032737	100000	001340	BIT	#ERR, \$RPCS	: ERROR ?	
3575	014512	001420			BEQ	58	: BR IF NOT	
3576	014514	005737	001314		TST	RETRY	: RETRY ATTEMPT ?	
3577	014520	001404			BEQ	38	: BR IF NOT	
3578	014522	032777	000040	164410	BIT	#SW05, JSWR	: DISPLAY ALL RETRY ATTEMPTS	
3579	014530	001401			BEQ	48	: BR IF NOT	
3580	014532	104062		38:	ERROR	62	: REPORT THE ERROR	
3581	014534	005237	001314	48:	INC	RETRY	: INCREMENT THE RETRY COUNT	
3582	014540	023737	001314	001316	CMP	RETRY, LRETRY	: RETRY LIMIT ?	
3583	014546	101734			BLOS	28	: BR IF NOT	
3584	014550	104110			ERROR	110	: REPORT UNSUCCESSFUL RETRY	
3585	014552	000410			BR	68	: CONTINUE	
3586	014554	005737	001314	58:	TST	RETRY	: COMING FROM A SUCCESSFUL RETRY ?	
3587	014560	001405			BEQ	68	: BR IF NOT	
3588	014562	112737	000111	001114	MOVB	#111, \$ITEMB	: RETRY COUNT MESSAGE NUMBER	
3589	014570	004737	017254		JSR	PC, TYPERR	: REPORT THE RETRY COUNT	
3590	014574	032777	000010	164336	68:	BIT	#SW03, JSWR	: COMPARE THE DATA ?
3591	014602	001002			BNE	78	: NO--BRANCH	
3592	014604	004737	024614		JSR	PC, DATCMP	: YES--DO IT	
3593	014610	004737	024114	78:	JSR	PC, INCADR	: INCREMENT THE DISK ADDRESS	
3594	014614	000703			BR	18	: CONTINUE	
3595	014616	000004		EXIT14:	SCOPE		: LOOP ?	
3596	014620	004737	023452		JSR	PC, RESLDR	: RESTORE THE LOADER IF IT WAS MOVED	
3597	014624	00C137	016616		JMP	SEOP	: GO TO END OF TEST	

```

*****
: *TEST 15          POWER FAIL TEST
*****
: *TEST THE ABILITY OF THE RP11E TO SENSE POWER FAILURE AND FOR THE DRIVES
: *TO RETRACT THEIR HEADS.  WHEN POWER IS TURNED ON AGAIN
: *THE CYLINDER ADDRESS IS TESTED FOR ZERO.  AFTER TYPING THE MESSAGE
: *REQUESTING POWER TO BE TURNED OFF, THE PROGRAM GOES INTO A LOOP

```

3600
3601
3602
3603
3604
3605

;*READING FROM THE SELECTED DISK DRIVE. AFTER POWER IS RESTORED,
;*MEMORY IS CHECKED TO SEE THAT THE DISK DID NOT TRANSFER ANY DATA
;*TO MEMORY WHILE POWER WAS GOING DOWN.

↑T15:

3606					BIT	BITS+(STN#2),TSTNMS	IS THIS TEST SELECTED
3607					BNE	+6	BR IF YES
3608					JMP	TST16	GO TO THE NEXT TEST
3609						#TST15,\$LPERR	SETUP THE ERROR LOOP ADDRESS
3610						#TST15,\$LPADR	SETUP THE SCOPE LOOP ADDRESS
3611	014630	033737	001432	001276	MOV	#15,\$STN#M	SETUP TEST NUMBER AND
3612	014630	001002			MOV	\$STN#M,\$DISP	CLEAR THE ERROR FLAG (\$ERFLG)
3613	014636	000137			MOV	#EXIT15,BYPASS	LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
3614	014640	012737			MOV	RPADR,R4	SETUP BYPASS ADDRESS
3615	014644	012737	001110		CLRB	DPB+1	RP11 BUS ADDRESS
3616	014652	012737	014630	001106	CLRB	DPB+3	CLEAR ANY EXTENDED MEMORY BITS
3617	014660	012737	000015	001102	TYPE	MSG15A	CLEAR 'MODE' & 'HDR' BITS
3618					JSR	PC,MOVLDR	'POWER FAIL TEST'
3619	014666	013777	001102	164246	MOV	(SP)+,LOMEM	MOVE THE LOADER
3620	014674	012737	015302	001200	CLR	LOMEM+2	BUFFER STARTING ADDRESS
3621	014702	013704	001324		MOV	#TACK,SP	CLEAR THE UPPER BITS JUST TO BE NICE
3622	014706	105037	002501		JSR	PC,CLRP	SETUP THE STACK POINTER
3623	014712	105037	002503		MOV	#TST15B,\$#24	CLEAR THE RP11
3624	014716	104401	027433		MOV	#PR7,\$#26	SET UP POWER FAIL VECTOR
3625	014722	004737	023346		MOV	LOMEM,R0	LOCKOUT INTERRUPTS
3626	014726	012637	001240		MOV	#125252,(R0)+	START OF FREE MEMORY SPACE
3627	014732	005037	001242		MOV	R0,MEMSIZ	FILL MEMORY WITH CHECKERBOARD
3628	014736	014736	001100		CMPL	\$	PATTERN
3629	014742	004737	025654		BLOS	\$	BR IF NOT FINISHED
3630	014746	012737	015314	000024	MOV	MAXCYL,\$CYL	LOAD MAXIMUM CYLINDER
3631	014754	012737	000340	000026	CLR	\$SEC	CLEAR SECTOR/TRACK
3632	014762	013700	001240		MOV	#256,\$WC	WORD COUNT = 1 SECTOR
3633	014766	012720	125252		MOV	LOMEM,\$BUF	BUFFER ADDRESS
3634	014772	020037	001302		MOV	#WRTSEK,DPB+2	WRITE OPERATION
3635	014776	101773			JSR	PC,RP11	START THE COMMAND
3636	015000	013737	001210	002510	JSR	PC,CONRDY	WAIT FOR THE CONTROLLER TO BE READY
3637	015006	005037	002512		JSR	PC,SAVRP	STORE THE REGISTERS
3638	015012	012737	000256	002504	BIT	#ERR,\$RPCS	ERROR DURING OPERATION
3639	015020	013737	001240	002506	BEG	\$	BR IF NOT
3640	015026	112737	000003	002502	ERROR	4	REPORT THE ERROR
3641	015034	004737	022770		TYPE	MSG15B	'TURN CPU POWER OFF AFTER MESSAGE'
3642	015040	004737	023664		JSR	PC,CLRP	CLEAR THE RP11
3643	015044	004737	023240		MOV	#RDSEK,DPB+2	READ COMMAND
3644	015050	032737	100000	001340	JSR	PC,RP11	INITIATE THE READ
3645	015056	001401			JSR	PC,CONRDY	WAIT FOR CONTROLLER READY
3646	015060	104004			BR	\$	LOOP, READING THE DISK
3647	015062	104401	027456				
3648	015066	004737	025654				
3649	015072	112737	000005	002502			
3650	015100	004737	022770				
3651	015104	004737	023664				
3652	015110	000766					

; AFTER MACHINE IS POWERED DOWN AND UP, CONTROL IS TRANSFERRED HERE.

3653					TST15A:	JSR	PC,RPINIT	INITIALIZE THE RP11E
3654						TYPE	MSG15D	'PROGRAM WILL LOOP, WAITING FOR DRIVE TO COME ONLINE'
3655								
3656	015112	004737	022566					
3657	015116	104401	027652					
3658	015122	012737	015122	001110	IS:	MOV	#15,\$LPERR	SETUP LOOP ON ERROR ADDRESS
3659	015130	012737	002500	001220		MOV	#2500,STALL	LOAD STALL VALUE
3660	015136	004737	023202			JSR	PC,STALL	STALL FOR 2500 MILLISECONDS
3661	015142	0137C1	0C1176			MOV	CHKDRV,R1	DRIVE ADDRESS

T15 POWER FAIL TEST

```

3662 015146 004737 022666 JSR PC, DRVINT ; CHECK THE DRIVE'S STATUS
3663 015152 105761 001370 ISIB DBVSTA(R1) ; IS THE DRIVE ONLINE ?
3664 015156 001761 BEQ 15 ; CONTINUE WAITING FOR DRIVE TO COME ONLINE
3665 015160 003002 BGT 25 ; BR IF IT IS ONLINE
3666 015164 104112 ERROR 112 ; DRIVE UNSAFE AFTER POWER FAIL
3667 015164 000446 BR EXIT15 ; DRIVE UNSAFE, EXIT
3668 015166 113764 001176 000005 25: MOVB CHKDRV, RPCS+1(R4) ; SELECT THE DRIVE
3669 015174 004737 023240 JSR PC, SAVAP ; STORE THE REGISTERS
3670 015200 005737 001354 TST %SUCA ; SELECTED CYLINDER SHOULD BE ZERO
3671 015204 001404 BEQ 35 ; BR IF IT DID
3672 015206 012737 015216 001164 MOV #35, %SECAPE ; ESCAPE TO 35 ON ERROR
3673 015214 104063 ERROR 63 ; DRIVE DID NOT RETURN TO CYLINDER C
3674 ; ON POWER FAILURE
3675 015216 35:
3676 015216 012737 015276 001164 MOV #65, %SECAPE ; ESCAPE TO 65 ON ERROR
3677 015224 013700 001240 MOV LOMEM, R0 ; BUFFER ADDRESS
3678 015230 022720 125252 45: CMP #125252, (R0)+ ; DID MEMORY RETAIN PATTERN ON POWER FAIL ?
3679 015234 001004 BNE 55 ; BRANCH IF NOT
3680 015236 020037 001302 CMP R0, MEMSIZ ; END OF MEMORY ?
3681 015242 001372 BNE 45 ; BR IF NOT
3682 015244 000414 BR 65 ; EXIT
3683 015246 012737 125252 001124 55: MOV #125252, %GDDAT ; EXPECTED MEMORY CONTENTS
3684 015254 016037 177776 001126 MOV -2(R0), %BDDAT ; ACTUAL MEMORY CONTENTS
3685 015262 010037 001122 MOV R0, %BDDADR ; ADDRESS
3686 015266 162737 000002 001122 SUB #2, %BDDADR ; CORRECT IT
3687 015274 104064 ERROR 64 ; CONTENTS OF MEMORY CHANGED AFTER POWER FAIL
3688 015276 104401 027617 65: TYPE .MSG15C ; 'END OF POWER FAIL TEST'
3689 015302 000004 EXIT15: SCOPE ; LOOP ?
3690 015304 004737 023452 JSR PC, RESLDR ; RESTORE THE LOADER
3691 015310 000137 003676 JMP START2 ; RETURN TO 'CONVERSATION MODE' ENTRANCE
3692 ; POWER FAIL TRAP HANDLER
3693
3694 015314 012737 015324 000024 TST15B: MOV #TST15C, @#24 ; SET UP POWER UP VECTOR
3695 015322 000774 BR TST15B ; LOOP
3696 ; POWER UP TRAP HANDLER
3697
3698
3699
3700 015324 012737 000026 000024 TST15C: MOV #26, @#24 ; RESTORE TRAP CATCHER
3701 015332 005037 000026 CLR @#26 ; CATCHER HALT
3702 015336 012706 001100 MOV #STACK, SP ; SETUP STACK POINTER
3703 015342 005037 177776 CLR PS ; RETURN TO PRIORITY ZERO
3704 015346 012737 011610 001220 MOV #5000, STALLT ; LOAD STALL VALUE
3705 015354 004737 023202 JSR PC, STALL ; STALL FOR 5000. MILLISECONDS
3706 015360 004737 020416 JSR PC, STKINT ; INITIALIZE THE TTY KEYBOARD
3707 015364 013704 001324 MOV RPADR, R4 ; RP11E ADDRESS
3708 015370 00065C BR TST15A ; VERIFY POWER DOWN SEQUENCE

```

```

*****
; *TEST 16 HEAD ALIGNMENT ROUTINE
; *THIS ROUTINE PERFORMS HEAD SELECTION AS DIRECTED FROM THE KEYBOARD.
; *THE ALIGNMENT OF THE SELECTED HEAD MAY BE CHECKED OR ADJUSTED.
*****

```

3709
3710
3711
3712
3713
3714
3715
3716
3717
3718
3719
3720
3721
3722
3723
3724
3725
3726
3727
3728
3729
3730
3731
3732
3733
3734
3735
3736
3737
3738
3739
3740
3741
3742
3743
3744
3745
3746
3747
3748
3749
3750
3751
3752
3753
3754
3755
3756
3757
3758
3759
3760
3761
3762
3763
3764
3765
3766
3767
3768
3769
3770
3771
3772
3773
3774
3775
3776
3777
3778
3779
3780
3781
3782
3783
3784
3785
3786
3787
3788
3789
3790
3791
3792
3793
3794
3795
3796
3797
3798
3799
3800


```

3718 015372 033737 001434 001276 TST16: BIT BITS+(&STN*2),TSTNMS :IS THIS TEST SELECTED
3719 015372 001002 BNE +6 :BR IF YES
3720 015400 000137 JMP TST17 :GO TO THE NEXT TEST
3721 015402 015624 MOV #TEST16,$LPERR :SETUP THE ERROR LOOP ADDRESS
3722 015406 012737 015460 001110 MOV #TST16,$LPADR :SETUP THE SCOPE LOOP ADDRESS
3723 015414 012737 015372 001106 MOV #16,$STNM :SETUP TEST NUMBER AND
3724 015422 012737 000016 001102 :CLEAR THE ERROR FLAG (SERFLG)
3725 015430 013777 001102 163504 MOV $STNM,$DISPLAY :LOAD THE TEST NUMBER INTO THE DISPLA REGISTER
3726 015436 012737 015620 001200 MOV #EXIT16,BYPASS :SETUP BYPASS ADDRESS
3727 015444 013704 001324 MOV RPADR,R4 :RP11E BUS ADDRESS
3728 015450 105037 002501 CLRB DPB+1 :CLEAR ANY EXTENDED MEMORY BITS
3729 015454 105037 002503 CLRB DPB+3 :CLEAR 'MODE' & 'HDR' BITS
3730 015454 012706 001100 TEST16: MOV #STACK,SP :SETUP THE STACK POINTER
3731 015460 104401 027743 TYPE MSG16A :'HEAD ALIGNMENT ROUTINE'
3732 015470 000410 BR 3$ :GET THE HEAD ADDRESS WITHOUT THE SW07 NONSENSE
3733 015472 032777 000200 163440 1$: BIT #SW07,$SWR :SWITCH 7 SET ?
3734 015500 001774 BEQ 1$ :BR IF NOT
3735 015502 032777 000200 163430 2$: BIT #SW07,$SWR :SWITCH 7 RESET ?
3736 015510 001374 BNE 2$ :BR IF NOT
3737 015512 104401 030255 3$: TYPE ,MSG16B :'ENTER HEAD ADDRESS (IN OCTAL)'
3738 015516 104411 ROLIN :GET THE ENTRY
3739 015520 012601 MOV (SP)+,R1 :STARTING ADDRESS OF INPUT ASCII STRING
3740 015522 004037 027136 JSR RO,CK.NUM :CHECK THE NUMBER
3741 015526 015512 JS 4$ :ILLEGAL INPUT
3742 015530 015534 JS 4$ :TERMINATED WITH A 'CR'
3743 015532 015512 JS 3$ :NO ENTRY - 'CR' ONLY
3744 015534 122702 000024 4$: CMPB #20.,R2 :VALID ?
3745 015540 101764 BLOS 3$ :BR IF NOT
3746 015542 110237 002513 MOVB #R2,$TRK :LOAD NEW HEAD ADDRESS
3747 015546 136237 001360 001304 BITB ATABIT(R2),DRVTYPE :RPO2 OR RPO3 ?
3748 015554 001004 BNE 5$ :BR IF RPO3
3749 015556 012737 000111 00251C MOV #73.,$CYL :RPO2 ALIGNMENT CYLINDER
3750 015564 000403 BR 6$ :CONTINUE
3751 015566 012737 000222 002510 5$: MOV #146.,$CYL :RPO3 ALIGNMENT CYLINDER
3752 015574 112737 000011 002502 6$: MOVB #SEEK,DPB+2 :SEEK COMMAND
3753 015602 004737 025654 JSR PC,CLAP :CLEAR THE RP11
3754 015606 004737 022770 JSR PC,RP11 :SEEK AND LOAD THE HEAD
3755 015612 004737 023546 JSR PC,DRVRDY :WAIT FOR THE DRIVE TO BECOME READY
3756 015616 000725 BR 1$ :WAIT FOR SWITCH 7 TO BE TOGGLED
3757 015620 000000 EXIT16: HALT :'EXIT' FOR COMPATABILITY WITH OTHER TESTS
3758 015622 000776 BR -2 :LOCK THE HALT

```

:TEST 17 RP11E/DRIVE CONTROL PANEL SWITCH TEST

:*THIS TEST TEST THE 'ENABLE/DISABLE' AND 'READ ONLY' SWITCHES ON
:*THE DRIVE CONTROL PANEL AND TESTS THE 'WRITE LOCKOUT' AND 'LOA'
:*SWITCHES ON THE RP11E/DRIVE CONTROL PANEL.

TST17:

```

3770 015624 033737 001436 001276 BIT BITS+(&STN*2),TSTNMS :IS THIS TEST SELECTED
3771 015624 001002 BNE +6 :BR IF YES
3772 015632 000137 JMP #SEOP :GO TO END OF TEST
3773 015634 000137 016616

```



```

3774 015640 012737 015716 001110      MOV      #TEST17,$LPERR      :SETUP THE ERROR LOOP ADDRESS
3775 015646 012737 015624 001106      MOV      #TST17,$LPADR      :SETUP THE SCOPE LOOP ADDRESS
3776 015654 012737 000017 001102      MOV      #17,$STSTNM        :SETUP TEST NUMBER AND
3777                                     :CLEAR THE ERROR FLAG ($ERFLG)
3778 015662 013777 001102 163252      MOV      $STSTNM,$DISPLAY    :LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
3779 015670 012737 016604 001200      MOV      #EXIT17,$BYPASS     :SETUP BYPASS ADDRESS
3780 015676 013704 001324             MOV      RPADR,R4           :RP11E BUS ADDRESS
3781 015702 105037 002501             CLR      DPB+1              :CLEAR ANY EXTENDED MEMORY BITS
3782 015706 105037 002503             CLR      DPB+3              :CLEAR 'MODE' & 'HDR' BITS
3783 015712 104401 030317             TYPE     MSG17A              :RP11E CONTROL PANEL SWITCH TEST
3784 015716 012706 001100      TEST17: MOV      #STACK,SP    :SETUP THE STACK POINTER

                                     :CHECK THE ENABLE/DISABLE SWITCH ON THE DRIVE
3788 015722 104401 030362             TYPE     MSG17B ;'TYPE A CR TO START TEST AFTER EACH SETUP'
3789 015726 104401 030441      2$: TYPE     MSG17C          : 'DISABLE DRIVE # '
3790 015732 013746 001176      MOV      CHKDRV,-(SP)       :SAVE CHKDRV FOR TYPEOUT
3791                                     :TYPE DRIVE NUMBER
3792                                     :GO TYPE--OCTAL ASCII
3793 015736 104403             TYPPOS   1                  :TYPE 1 DIGIT(S)
3794 015740             .BYTE    1                  :TYPE LEADING ZEROS
3795 015741             .BYTE    1                  :TYPE LEADING ZEROS
3796 015742 104411             RDLIN    1                  :WAIT FOR THE 'CR'
3797 015744 012737 015744 001.10      3$: MOV      #3$, $LPERR      :CHANGE LOOP ON ERROR ADDRESS
3798 015752 004737 025654             JSR      PC,$CLRP           :CLEAR THE RP11
3799 015756 113764 001176 000005      MOV      CHKDRV,RPCS+1(R4)   :SELECT DRIVE
3800 015764 004737 023240             JSR      PC,$SARVP          :SAVE THE RP11 REGISTERS
3801 015770 032737 040000 001334      BIT      #SUOL,$RPO5        :IS THE UNIT ON LINE?
3802 015776 001401             BEQ      4$                 :BRANCH IF NOT
3803 016000 104065             ERROR    65                 :SELECTED UNIT ON LINE WITH THE
3804                                     :DRIVE DISABLED
3805 016002 104401 030464      4$: TYPE     MSG17D          : 'ENABLE DRIVE # '
3806 016006 013746 001176      MOV      CHKDRV,-(SP)       :SAVE CHKDRV FOR TYPEOUT
3807                                     :TYPE DRIVE NUMBER
3808                                     :GO TYPE--OCTAL ASCII
3809 016012 104403             TYPPOS   1                  :TYPE 1 DIGIT(S)
3810 016014             .BYTE    1                  :TYPE LEADING ZEROS
3811 016015             .BYTE    1                  :TYPE LEADING ZEROS
3812 016016 104411             RDLIN    1                  :WAIT FOR THE 'CR'
3813 016020 012737 016020 001110      5$: MOV      #5$, $LPERR      :CHANGE THE LOOP ON ERROR ADDRESS
3814 016026 004737 025654             JSR      PC,$CLRP           :CLEAR THE RP11
3815 016032 113764 001176 000005      MOV      CHKDRV,RPCS+1(R4)   :RESELECT THE UNIT
3816 016040 004737 023240             JSR      PC,$SARVP          :SAVE THE RP11 REGISTERS
3817 016044 032737 040000 001334      BIT      #SUOL,$RPO5        :IS THE UNIT ON LINE?
3818 016052 001001             BNE     RDSW                :BR IF ONLINE
3819 016054 104066             ERROR    66                 :SELECTED UNIT NOT ON LINE WITH
3820                                     :THE DRIVE ENABLED

                                     :CHECK THE DRIVE'S WRITE LOCKOUT SWITCH
3822 016056 104401 030506      RDSW: TYPE     MSG17E          : 'SET READ ONLY ON DRIVE '
3823 016062 013746 001176      MOV      CHKDRV,-(SP)       :SAVE CHKDRV FOR TYPEOUT
3824                                     :TYPE DRIVE NUMBER
3825                                     :GO TYPE--OCTAL ASCII
3826 016066 104403             TYPPOS   1                  :TYPE 1 DIGIT(S)
3827 016070             .BYTE    1                  :TYPE LEADING ZEROS
3828 016071             .BYTE    1                  :TYPE LEADING ZEROS
3829 016072 104411             RDLIN    1                  :WAIT FOR 'CR'
3830 016074 012737 016074 001110      2$: MOV      #2$, $LPERR      :LOAD LOOP ON ERROR ADDRESS

```

```

3830 016102 004737 025654 JSR PC,CLRP ;CLEAR THE RP11
3831 016106 113764 001176 000005 MOVB CHKDRV,RPCS+1(R4) ;SELECT THE DRIVE
3832 016114 004737 023240 JSR PC,SAVRP ;SAVE THE RP11 REGISTERS
3833 016120 032737 000400 001334 BIT #SUWP,$RPDS ;IS SELECTED UNIT WRITE PROTECTED?
3834 016126 001001 BNE 38 ;BR IF NOT
3835 016130 104067 ERROR 67 ;SELECTED UNIT WRITE PROTECT DIC
3836 ;NOT SET WITH READ ONLY SET
3837 016132 104401 030542 38: TYPE MSG17F ;SET READ/WRITE ON DRIVE
3838 016136 013746 001176 MOV CHKDRV,-(SP) ;SAVE CHKDRV FOR TYPEOUT
3839 ;TYPE DRIVE NUMBER
3840 016142 104403 TYPOS ;GO TYPE--OCTAL ASCII
3841 016144 001 .BYTE 1 ;TYPE 1 DIGIT(S)
3842 016145 001 .BYTE 1 ;TYPE LEADING ZEROS
3843 016146 104411 ROLIN ;WAIT FOR 'CR'
3844 016150 012737 016150 001110 48: MOV #48,$LPERR ;CHANGE THE LOOP ON ERROR ADDRESS
3845 016156 004737 025654 JSR PC,CLRP ;CLEAR THE RP11
3846 016162 113764 001176 000005 MOVB CHKDRV,RPCS+1(R4) ;RESELECT THE DRIVE
3847 016170 004737 023240 JSR PC,SAVRP ;SAVE THE RP11 REGISTERS
3848 016174 032737 000400 001334 BIT #SUWP,$RPDS ;IS SELECTED UNIT WRITE PROTECTED?
3849 016202 001401 BEQ WRSW ;BR IF NOT
3850 016204 104070 ERROR 70 ;SELECTED UNIT WRITE PROTECT SET
3851 ;WITH DRIVE WRITE ENABLED
3852
3853 ;TEST THE SETTING OF WRITE LOCKOUT AND THE 'LOA' SWITCHES ON THE
3854 ;RP11E CONTROL PANEL.
3855
3856 016206 104401 030577 WRSW: TYPE .MSG17G ;'SET WRITE LOCKOUT' ETC
3857 016212 104411 ROLIN ;WAIT FOR 'CR' FROM THE OPERATOR
3858 016214 012737 016214 001110 18: MOV #18,$LPERR ;CHANGE LOOP ON ERROR ADDRESS
3859 016222 005064 000012 CLR RPC(R4) ;CLEAR THE CYLINDER REGISTER
3860 016226 004737 023240 JSR PC,SAVRP ;SAVE THE RP11 REGISTERS
3861 016232 032737 000400 001334 BIT #SUWP,$RPDS ;IS SELECTED UNIT WRITE PROTECT SET?
3862 016240 001001 BNE 28 ;BR IF SET
3863 016242 104071 ERROR 71 ;SELECTED UNIT WRITE PROTECT SHOULD
3864 ;BE SET - RPCA EQUAL TO ZERO
3865 016244 012764 000002 000012 28: MOV #2,RPC(R4) ;LOAD A 2 INTO RPCA
3866 016252 004737 023240 JSR PC,SAVRP ;SAVE THE RP11 REGISTERS
3867 016256 032737 000400 001334 BIT #SUWP,$RPDS ;IS SELECTED UNIT WRITE PROTECTED?
3868 016264 001401 BEQ WRSW1 ;BR IF NOT PROTECTED
3869 016266 104072 ERROR 72 ;SELECTED UNIT WRITE PROTECT IS SET
3870 ;WITH EQUAL TO 2.
3871
3872 ;CHECK THE CYLINDER 'LOA' SWITCHES
3873
3874 016270 012700 000002 WRSW1: MOV #2,R0 ;INITIALIZE SWITCH PATTERN
3875 016274 104401 030714 TYPE .MSG17H ;'SET FOLLOWING VALUES IN THE
3876 ;CYLINDER LOA SWITCHES'
3877 016300 104401 031014 18: TYPE MSG17I ;'SET'
3878 016304 006200 ASR R0 ;SETUP PATTERN FOR TYPOUT
3879 016306 010046 MOV R0,-(SP) ;SAVE R0 FOR TYPEOUT
3880 ;TYPE THE SWITCH VALUE
3881 016310 104403 TYPOS ;GO TYPE--OCTAL ASCII
3882 016312 003 .BYTE 3 ;TYPE 3 DIGIT(S)
3883 016313 001 .BYTE 1 ;TYPE LEADING ZEROS
3884 016314 006300 ASL R0 ;RESTORE THE PATTERN
3885 016316 104411 ROLIN ;WAIT FOR 'CR' FROM THE OPERATOR

```

```

3886 016320 012737 016320 001110 2S:  MOV      #2$, $LPERR      ;CHANGE LOOP ON ERROR ADDRESS
3887 016326 010064 000012          MOV      R0, RPCA(R4)    ;LOAD PATTERN INTO RPCA
3888 016332 004737 023240          JSR      PC, SAVRP      ;SAVE THE RP11 REGISTERS
3889 016336 032737 000400 001334  BIT      #SUWP, $RPDS    ;IS UNIT WRITE PROTECT SET ?
3890 016344 001001          BNE     3891            ;BR IF SET
3891 016346 104073          ERROR   73             ;UNIT WRITE PROTECT IS NOT SET
3892                                     ;RPCA EQUALS CONTENTS OF BOUNDARY
3893                                     ;SWITCHES
3894 016350 062764 000002 000012 3S:  ADD      #2, RPCA(R4)    ;INCREMENT CYLINDER ADDRESS
3895 016356 012737 016356 001110 4S:  MOV      #4$, $LPERR    ;CHANGE THE LOOP ON ERROR ADDRESS
3896 016364 004737 023240          JSR      PC, SAVRP      ;SAVE THE RP11 REGISTERS
3897 016370 032737 000400 001334  BIT      #SUWP, $RPDS    ;IS UNIT WRITE PROTECT SET ?
3898 016376 001401          BEQ     55             ;BR IF NOT
3899 016400 104074          ERROR   74             ;UNIT WRITE PROTECT IS SET WITH
3900                                     ;RPCA ONE GREATER THAN BOUNDARY
3901                                     ;SWITCH SETTINGS
3902 016402 006300          5S:  ASL      R0              ;UPDATE TEST PATTERN
3903 016404 032700 001000          BIT      #BIT09, R0     ;IS PATTERN EXCEEDED?
3904 016410 001733          BEQ     15             ;BRANCH IF NOT
3905
3906                                     ;CHECK DRIVE 'LOA' SWITCHES
3907
3908 016412 005037 001176  WRSW2: CLR      CHKDRV      ;START WITH DRIVE 0
3909 016416 104401 031037          TYPE   ,MSG17J         ;RESET THE CYLINDER LOA SWITCHES
3910                                     ;SET THE FOLLOWING VALUES IN THE DRIVE
3911                                     ;'LOA SWITCHES'
3912 016422 012764 000002 000012 1S:  MOV      #2, RPCA(R4)    ;LOAD CYLINDER ADDRESS
3913 016430 113764 001176 000005  MOVVB   CHKDRV, RPCS+1(R4) ;SELECT THE FIRST DRIVE NUMBER
3914 016436 104401 031014          TYPE   ,MSG17I         ;SET SWITCHES TO '
3915 016442 013746 001176          MOV     CHKDRV, -(SP)   ;SAVE CHKDRV FOR TYPEOUT
3916                                     ;TYPE SWITCH SETTING
3917 016446 104403          TYPOS   ;GO TYPE--OCTAL ASCII
3918 016450 001          .BYTE  1              ;TYPE 1 DIGIT(S)
3919 016451 001          .BYTE  1              ;TYPE LEADING ZEROS
3920 016452 104411          ROLIN   ;WAIT FOR THE OPERATOR TO ENTER A 'CR'
3921 016454 012737 016454 001110 2S:  MOV      #2$, $LPERR    ;LOOP ON ERROR ADDRESS
3922 016462 004737 023240          JSR      PC, SAVRP      ;STORE THE RP11 REGISTERS
3923 016466 032737 000400 001334  BIT      #SUWP, $RPDS    ;IS THE SELECTED DRIVE WRITE PROTECTED ?
3924 016474 001401          BEQ     35             ;BR IF NOT
3925 016476 104075          ERROR   75             ;'SUWP' SET WHEN DRIVE LOA SWITCHES
3926                                     ;EQUAL THE SELECTED DRIVE'
3927 016500 005737 001176 3S:  TST     CHKDRV         ;THROUGH HERE THE FIRST TIME ?
3928 016504 001004          BNE     45             ;BR IF NOT
3929 016506 012737 000001 001176  MOV     #1, CHKDRV      ;SET FOR DRIVE 1
3930 016514 000406          BR      55             ;CONTINUE
3931 016516 006337 001176 4S:  ASL     CHKDRV         ;SHIFT TEST ADDRESS
3932 016522 032737 000010 001176  BIT     #BIT03, CHKDRV   ;FINISHED WITH ADDRESSES ?
3933 016530 001025          BNE     EXIT17         ;BR IF FINISHED
3934 016532 104401 031014 5S:  TYPE   ,MSG17I         ;SET SWITCHES TO '
3935 016536 013746 001176          MOV     CHKDRV, -(SP)   ;SAVE CHKDRV FOR TYPEOUT
3936                                     ;TYPE SWITCH SETTING
3937 016542 104403          TYPOS   ;GO TYPE--OCTAL ASCII
3938 016544 001          .BYTE  1              ;TYPE 1 DIGIT(S)
3939 016545 001          .BYTE  1              ;TYPE LEADING ZEROS
3940 016546 104411          ROLIN   ;WAIT FOR THE OPERATOR TO ENTER A 'CR'
3941 016550 012737 016550 001110 6S:  MOV      #6$, $LPERR    ;CHANGE ERROR LOOP ADDRESS

```

```

3942 016556 004737 023240 JSR PC,SAVRP ;STORE THE REGISTERS
3943 016562 032737 000400 001334 BIT #SUWP,$RPOD ;IS THE DRIVE WRITE PROTECTED ?
3944 016570 001001 BNE 7$ ;BR IF IT IS
3945 016572 104113 ERROR 113 ;'SUWP' NOT SET WHEN DRIVE LOA SWITCHES
;GREATER THAN SELECTED DRIVE NUMBER
3946
3947 016574 113764 001176 000005 7$: MOVB CHKDRV,$RPOD+1(R4) ;SELECT THE NEXT DRIVE
3948 016602 000724 BR 2$ ;CONTINUE WITH THE TEST
3949 016604 000004 EXIT17: SCOPE ;LOOP ?
3950 016606 104401 031176 TYPE MSG17K ;END OF CONTROL PANEL TEST - RETURN SWITCHES TO NORMAL
3951 016612 000137 003676 JMP START2 ;RETURN TO THE 'CONVERSATION MODE' ENTRANCE
3952
3953 .SBTTL END OF PASS ROUTINE
3954
3955 ;*****
3956 ;* INCREMENT THE PASS NUMBER ($PASS)
3957 ;* INDICATE END-OF-PROGRAM AFTER 1 PASSES THRU THE PROGRAM
3958 ;* IF THERES A MONITOR GO TO IT
3959 ;* IF THERE ISN'T JUMP TO RESTART
3960
3961 016616 SEOP: TYPE 65$ ;:TYPE ASCIZ STRING
3962 016616 104401 016624 BR 64$ ;:GET OVER THE ASCIZ
3963 016622 00041C ;:65$: .ASCIZ '<15><12><12> END OF PASS/'
3964
3965 016644 64$: TST 2#DRVSEL ;:ANY DRIVES SELECTED
3966 016644 005737 001312 BEQ 1$ ;:NO, BRANCH
3967 016650 001436 TYPE 67$ ;:TYPE ASCIZ STRING
3968 016652 104401 016660 BR 66$ ;:GET OVER THE ASCIZ
3969 016656 000407 ;:67$: .ASCIZ '/ USING DRIVE/'
3970
3971 016676 66$: MOV 2#CHKDRV,-(SP) ;:SAVE 2#CHKDRV FOR TYPEOUT
3972 016676 013746 001176 TYPOS ;:GO TYPE--OCTAL ASCII .
3973 016702 104403 .BYTE 2 ;:TYPE 2 DIGIT(S)
3974 016704 002 .BYTE 0 ;:SUPPRESS LEADING ZEROS
3975 016705 000 TYPE 69$ ;:TYPE ASCIZ STRING
3976 016706 104401 016714 BR 68$ ;:GET OVER THE ASCIZ
3977 016712 000412 ;:69$: .ASCIZ '/ ERRORS DETECTED='
3978
3979 016740 68$: MOV 2#SERTTL,-(SP) ;:SAVE 2#SERTTL FOR TYPEOUT
3980 016740 013746 001112 TYPOC ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
3981 016744 104402 1$: CLR 2#SERTTL ;:ZERO ERROR TOTAL
3982 016746 005037 001112 CLR $STNM ;:ZERO THE TEST NUMBER
3983 016752 005037 001102 CLR $TIMES ;:ZERO THE NUMBER OF ITERATIONS
3984 016756 005037 001162 INC $PASS ;:INCREMENT THE PASS NUMBER
3985 016762 005237 001100 BIC #10000,$PASS ;:DON'T ALLOW A NEG. NUMBER
3986 016766 042737 100000 001100 DEC (PC)+ ;:LOOP?
3987 016774 005327 SEOPCT: .WORD 1
3988 016776 000001 BGT $DOAGN ;:YES
3989 017000 003027 MOV (PC)+,2(PC)+ ;:RESTORE COUNTER
3990 017002 012737 SENDCT: .WORD 1
3991 017004 000001 SEOPCT
3992 017006 016776 TYPE 65$ ;:TYPE ASCIZ STRING
3993 017010 104401 017016 BR 64$ ;:GET OVER THE ASCIZ
3994 017014 000407 ;:65$: .ASCIZ '/15<12>/END OF TEST/'
3995
3996 017034 64$: TYPE ,SNULL ;:TYPE NULL CHARACTER
3997 017034 104401 017064

```

END OF PASS ROUTINE

```

3998 017040 013700 000042 $GET42: MOV 2#42,RO ;: GET MONITOR ADDRESS;
3999 017044 001405 BEQ $DOAGN ;: BRANCH IF NO MONITOR
4000 017046 000005 RESET ;: CLEAR THE WORLD
4001 017050 004710 $ENDAD: JSR PC,(RO) ;: GO TO MONITOR
4002 017052 000240 NOP ;: SAVE ROOM
4003 017054 000240 NOP ;: FOR
4004 017056 000240 NOP ;: ACT11
4005 017060 $DOAGN: ;:
4006 017060 000137 JMP 2(PC)+ ;: RETURN
4007 017062 004762 $RTNAD: .WORD RESTART
4008 017064 377 000 $ENULL: .BYTE -1,-1,0 ;: NULL CHARACTER STRING
4009 017070 .EVEN

```

.SBTTL *** SUBROUTINES ***

.SBTTL ERROR HANDLER ROUTINE

```

;: *****
;: *THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT.
;: *SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
;: *AND GO TO TYPERR ON ERROR
;: *THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
;: *SW15=1 HALT ON ERROR
;: *SW13=1 INHIBIT ERROR TYPEOUTS
;: *SW10=1 BELL ON ERROR
;: *SW09=1 LOOP ON ERROR
;: *CALL
;: * ERROR N ;: ERROR=EMT AND N=ERROR ITEM NUMBER

```

```

4029 017070 $ERROR:
4030 017070 104407 CKSWR ;: TEST FOR CHANGE IN SOFT-SWR
4031 017072 105237 001103 7$: INCB $ERFLG ;: SET THE ERROR FLAG
4032 017076 001775 BEQ 7$ ;: DON'T LET THE FLAG GO TO ZERO
4033 017100 013777 001102 162034 MOV $TSTNM,$DISP ;: DISPLAY TEST NUMBER AND ERROR FLAG
4034 017106 032777 002000 162024 BIT #BIT10,$SWR ;: BELL ON ERROR?
4035 017114 001402 BEQ 1$ ;: NO - SKIP
4036 017116 104401 001166 TYPE $BELL ;: RING BELL
4037 017122 005237 001112 1$: INC $ERTTL ;: COUNT THE NUMBER OF ERRORS
4038 017126 011637 001116 MOV (SP), $ERRPC ;: GET ADDRESS OF ERROR INSTRUCTION
4039 017132 162737 000002 001116 SUB #2, $ERRPC
4040 017140 117737 161752 001114 MOV $ERRPC, $ITEMB ;: STRIP AND SAVE THE ERROR ITEM CODE
4041 017146 032777 020000 161764 BIT #BIT13,$SWR ;: SKIP TYPEOUT IF SET
4042 017154 001004 BNE 20$ ;: SKIP TYPEOUTS
4043 017156 004737 017254 JSR PC, TYPERR ;: GO TO USER ERROR ROUTINE
4044 017162 104401 001173 TYPE , $CRLF
4045 017166 20$:
4046 017166 005777 161746 2$: TST $SWR ;: HALT ON ERROR
4047 017172 100002 BPL 3$ ;: SKIP IF CONTINUE
4048 017174 000000 HALT ;: HALT ON ERROR!
4049 017176 104407 CKSWR ;: TEST FOR CHANGE IN SOFT-SWR
4050 017200 032777 001000 161732 3$: BIT #BIT09,$SWR ;: LOOP ON ERROR SWITCH SET?
4051 017206 001402 BEQ 4$ ;: BR IF NO
4052 017210 013716 001110 MOV $LPERR,(SP) ;: FUDGE RETURN FOR LOOPING
4053 017214 005737 001164 4$: TST $ESCAPE ;: CHECK FOR AN ESCAPE ADDRESS

```

ERROR HANDLER ROUTINE

```

4100 017220 001402          BEQ      55          ;; BR IF NONE
4101 017222 013716 001164    MOV      55         SEESCAPE, (SP)    ;; FUDGE RETURN ADDRESS FOR ESCAPE
4102 017224 022737 017050 000042 55:      CMP      65         #SENDAD, 2042    ;; ACT-11 AUTO-ACCEPT?
4103 017226 001001          BNE     65         65          ;; BRANCH IF NO
4104 017228 000000          HALT                    ;; YES
4105 017230 023737 000042 000046 65:      CMP      125        2042, 2046          ;; ACT11 AUTO MODE?
4106 017232 001001          BNE     125         125          ;; NO RETURN
4107 017234 000000          HALT                    ;; HALT ON ERROR
4108 017236 000002          RTI                     ;; RETURN
4109 C:740E 005705

;*****
; THIS ROUTINE USES THE "ITEM CONTROL BYTE" (SITEMB) TO DETERMINE
; WHICH ERROR IS TO BE REPORTED. IT THEN OBTAINS FROM THE "ERROR
; TABLE" (SERRTB) AND REPORTS THE APPROPRIATE INFORMATION
; CONCERNING THE ERROR.
; CALL
;     JSR      PC, TYPERR
;     RETURN
;
; TYPERR: SAVREG          ;; SAVE R0-R5
;         BIT      #SW13, 25MR    ;; INHIBIT TIMEOUTS ?
;         BNE     20$           ;; BR IF YES
;         CLR     STMPD          ;; CLEAR LOCATION FOR TEST NUMBER
;         MOVB   STSTNM, STMPD   ;; LOAD TEST NUMBER FOR TYPEOLT
;         CLR     R0            ;; CLEAR R0 FOR ERROR NUMBER
;         MOVB   SITEMB, R0     ;; ERROR NUMBER
;         DEC    R0            ;; FORM INDEX FOR ERROR TABLE
;         ASL   R0
;         ASL   R0
;         ASL   R0
;         ASL   R0
;         ASL   R0
;         ASL   R0
;         ASL   R0
;         ASL   R0
;         MOV    #SERRTB, R0    ;; FORM ADDRESS
;         (R0)+, 2$           ;; GET ERROR MESSAGE (EM) POINTER
;         BEQ    3$           ;; BRANCH IF THERE ISN'T ONE
;         TYPE  , $CRLF        ;; CARRIAGE RETURN - LINE FEED
;
;         .WORD  0             ;; "EM" POINTER GOES HERE
;         MOV    (R0)+, 4$     ;; PICK UP DATA HEADER (DH) POINTER
;         BEQ    5$           ;; BRANCH IF NONE
;         TYPE  , $CRLF        ;; CARRIAGE RETURN-LINE FEED
;
;         .WORD  0             ;; "DH" POINTER GOES HERE
;         MOV    (R0)+, R1     ;; PICKUP DATA TABLE (DT) POINTER
;         BEQ    20$          ;; BRANCH IF NONE
;         CLR   R5            ;; SET INDENT SWITCH
;         MOV   (R0)+, R0     ;; DATA FORMAT (DF) POINTER
;         MOV   (R0)+, R2     ;; NUMBER OF DH'S TO TYPE
;         BEQ   20$          ;; BRANCH IF DH NUMBER IS 0
;         COM   R5            ;; NO INDENT
;         MOVB (R0)+, R3     ;; NUMBER OF DATA WORDS TO TYPE
;         BEQ  20$           ;; BR IF NO WORDS TO TYPE
;         MOVB (R0)+, R4     ;; AND HOW TO TYPE THEM
;         TYPE  , $CRLF        ;; CR-LF
;         TST   R5            ;; INDENT ?

```

```

110 01710 001002      BNE 118      ;BR IF NOT
111 01711 104401 027430  TYPE BLNK52  ;BLANKS
112 01712 006004 118: ROR R4      ;OCTAL OR DECIMAL?
113 01713 103403      BCS 128     ;DECIMAL--BRANCH
114 01714 013146     MOV 2(R1)+,-(SP) ;SAVE 2(R1)+ FOR TYPEOUT
115 01715 104402     TYP0C      ;GO TYPE--OCTAL ASCII(ALL DIGITS)
116 01716 000702     BR 138
117 01717 013146     MOV 2(R1)+,-(SP) ;SAVE 2(R1)+ FOR TYPEOUT
118 01718 104405     TYPDS      ;GO TYPE--DECIMAL ASCII WITH SIGN
119 01719 005303 138: R3      ;MORE NUMBERS TO TYPE?
120 01720 001403     BEQ 148     ;NO--BRANCH
121 01721 104401 027430  TYPE BLNK52  ;YES--TYPE SEPARATORS
122 01722 000764     BR 118     ;LOOP
123 01723 005302 148: DEC R2      ;MORE OH'S?
124 01724 003713     BLE 208     ;NO--BRANCH
125 01725 104401 001173  TYPE SCRLF   ;YES--START A NEW LINE
126 01726 005105     COM R5      ;INDENT?
127 01727 001002     BNE 158     ;NO--BRANCH
128 01728 104401 027430  TYPE BLNK52  ;YES--TYPE SPACES
129 01729 012037 017474 158: MOV 1(R0)+,168 ;GET NEXT OH
130 01730 104401     TYPE      ;AND TYPE IT
131 01731 000000 168: .WORD 0    ;OH POINTER GOES HERE
132 01732 000736     BR 108     ;LOOP
133 01733 104413 208: RESREG    ;RESTORE RO-R5
134 01734 000207     RTS PC     ;RETURN

```

.SBTTL TYPE ROUTINE

```

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

```

```

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

```

```

154 017504 105737 001157 $TYPE: TSTB $TFPLG ;IS THERE A TERMINAL?
155 017510 100002      BPL 15      ;BR IF YES
156 017512 000000      HALT      ;HALT HERE IF NO TERMINAL
157 017514 000407     BR 35      ;LEAVE
158 017516 010046 15: MOV R0,-(SP) ;SAVE R0
159 017520 017600 000002  MOV 22(SP),R0 ;GET ADDRESS OF ASCIZ STRING
160 017524 112046 25: MOV8 (R0)+,-(SP) ;PUSH CHARACTER TO BE TYPED ONTO STACK
161 017526 001005     BNE 45      ;BR IF IT ISN'T THE TERMINATOR
162 017530 005726     TST (SP)+   ;IF TERMINATOR POP IT OFF THE STACK
163 017532 012600 60S: MOV (SP)+,R0 ;RESTORE R0
164 017534 062716 000002 35: ADD 2,(SP) ;ADJUST RETURN PC
165 017540 000002     RTI

```

```

4166 017542 122716 000011 4S:  CMPB  #HT,(SP)  ;;BRANCH IF (HT)
4167 017546 001430  BEQ  BS  ;;
4168 017550 122716 000200  CMPB  #CR,LF,(SP)  ;;BRANCH IF NOT (CR,LF)
4169 017554 001006  BNE  BS  ;;
4170 017556 005726  TST  (SP)+  ;;POP (CR)<LF> EQUIV
4171 017560 104401  TYPE  ;;TYPE A CR AND LF
4172 017562 001173  $CR,LF  ;;
4173 017566 105037 017720  CLRB  $CHARCNT  ;;CLEAR CHARACTER COUNT
4174 017568 000735  BR  BS  ;;GET NEXT CHARACTER
4175 017570 004737 017654  JSR  PC,$TYPEC  ;;GO TYPE THIS CHARACTER
4176 017572 123726 001156 6S:  $FILLC,(SP)+  ;;IS IT TIME FOR FILLER CHARS.?
4177 017574 001350  BNE  BS  ;;IF NO GO GET NEXT CHAR.
4178 017604 013746 001154  MOV  $NULL,-(SP)  ;;GET # OF FILLER CHARS. NEEDED
4179 017610 105366 000001 7S:  DEC  1(SP)  ;;DOES A NULL NEED TO BE TYPED?
4180 017614 002770  BLY  BS  ;;BR IF NO--GO POP THE NULL OFF OF STACK
4181 017616 004737 017654  JSR  PC,$TYPEC  ;;GO TYPE A NULL
4182 017618 105337 017720  DEC  $CHARCNT  ;;DO NOT COUNT AS A COUNT
4183 017626 000770  BR  BS  ;;LOOP

```

:HORIZONTAL TAB PROCESSOR

```

4184 017630 112716 000040 8S:  MOV  #' (SP)  ;;REPLACE TAB WITH SPACE
4185 017634 004737 017654 9S:  JSR  PC,$TYPEC  ;;TYPE A SPACE
4186 017640 132737 000007 017720  BITB  #',$CHARCNT  ;;BRANCH IF NOT AT
4187 017646 001372  BNE  BS  ;;TAB STOP
4188 017650 005726  TST  (SP)+  ;;POP SPACE OFF STACK
4189 017652 000724  BR  BS  ;;GET NEXT CHARACTER
4190 017654 105777 161270  $TYPEC:  TST  $STPB  ;;WAIT UNTIL PRINTER IS READY
4191 017660 100375  BPL  $TYPEC  ;;
4192 017662 116677 000002 161262  MOV  2(SP),2$STPB  ;;LOAD CHAR TO BE TYPED INTO DATA REG.
4193 017670 122766 000015 000002  CMPB  #CR,2(SP)  ;;IS CHARACTER A CARRIAGE RETURN?
4194 017676 001003  BNE  BS  ;;BRANCH IF NO
4195 017700 105037 017720  CLRB  $CHARCNT  ;;YES--CLEAR CHARACTER COUNT
4196 017704 000406  BR  $TYPEX  ;;EXIT
4197 017706 122766 000012 000002 1S:  CMPB  #LF,2(SP)  ;;IS CHARACTER A LINE FEED?
4198 017714 001402  BEQ  $TYPEX  ;;BRANCH IF YES
4199 017716 105227  INCB  (PC)+  ;;COUNT THE CHARACTER
4200 017720 000000  $CHARCNT:  WORD  0  ;;CHARACTER COUNT STORAGE
4201 017722 000207  $TYPEX:  RTS  PC

```

.SBTTL BINARY TO OCTAL (ASCII) AND TYPE

```

4202 *****
4203 *THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
4204 *OCTAL (ASCII) NUMBER AND TYPE IT.
4205 *$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
4206 *CALL:
4207 *
4208 *   MOV  NUM,-(SP)  ;;NUMBER TO BE TYPED
4209 *   TYPOS  ;;CALL FOR TYPEOUT
4210 *   .BYTE  N  ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
4211 *   .BYTE  M  ;;M=1 OR 0
4212 *
4213 *   ;;1=TYPE LEADING ZEROS
4214 *   ;;0=SUPPRESS LEADING ZEROS

```


E07

```

*STYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
*STYPOS OR STYPOC
*CALL:
*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
*      TYPON                    ;;CALL FOR TYPEOUT
*
*STYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
*CALL:
*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
*      TYPOC                    ;;CALL FOR TYPEOUT
*
017724 017646 000000 020147 STYPOS: MOV      2(SP),-(SP)      ;; PICKUP THE MODE
017730 116637 000001 020147 MOVVB   1(SP),%SOFILL    ;; LOAD ZERO FILL SWITCH
017736 112637 020151 020151 MOVVB   (SP)+,%SOMODE+1  ;; NUMBER OF DIGITS TO TYPE
017742 062716 000002 020146 ADD      #2,(SP)        ;; ADJUST RETURN ADDRESS
017746 000406 020147 STYPOC: MOVVB   #1,%SOFILL    ;; SET THE ZERO FILL SWITCH
017750 112737 000001 020147 MOVVB   #6,%SOMODE+1    ;; SET FOR SIX(6) DIGITS
017756 112737 000006 020151 STYPON: MOVVB   #5,%SOCNT    ;; SET THE ITERATION COUNT
017764 112737 000005 020146 MOV      R3,-(SP)        ;; SAVE R3
017772 010346 020146 MOV      R4,-(SP)        ;; SAVE R4
017774 010446 020146 MOV      R5,-(SP)        ;; SAVE R5
017776 010546 020151 MOVVB   %SOMODE+1,R4    ;; GET THE NUMBER OF DIGITS TO TYPE
020000 113704 000006 020150 ADD      #6,R4          ;; SUBTRACT IT FOR MAX. ALLOWED
020004 005404 020147 MOVVB   R4,%SOMODE    ;; SAVE IT FOR USE
020006 062704 020147 MOVVB   %SOFILL,R4      ;; GET THE ZERO FILL SWITCH
020012 110437 020147 MOV      12(SP),R5      ;; PICKUP THE INPUT NUMBER
020016 113704 000012 020022 CLR      R5            ;; CLEAR THE OUTPUT WORD
020022 016605 000012 020026 ROL     R5            ;; ROTATE MSB INTO "C"
020026 005003 020030 1S:   ROL     R5            ;; GO DO MSB
020030 006105 020032 2S:   ROL     R5            ;; FORM THIS DIGIT
020032 000404 020034 3S:   ROL     R5            ;; GET LSB OF THIS DIGIT
020034 006105 020036 4S:   ROL     R5            ;; TYPE THIS DIGIT
020036 006105 020040 5S:   ROL     R5            ;; BR IF NO
020040 006105 020042 6S:   MOV     R5,R3          ;; GET RID OF JUNK
020042 010503 020044 7S:   DECIB  %SOMODE        ;; TEST FOR 0
020044 006103 020046 8S:   BPL     7S            ;; SUPPRESS THIS 0?
020046 105337 020150 9S:   BIC     #177770,R3    ;; BR IF YES
020052 100016 020054 10S:  BNE     4S            ;; DON'T SUPPRESS ANYMORE 0'S
020054 042703 17777C 11S:  TST     R4            ;; MAKE THIS DIGIT ASCII
020060 001002 020062 12S:  BEQ     5S            ;; MAKE ASCII IF NOT ALREADY
020062 005704 020064 13S:  INC     R4            ;; SAVE FOR TYPING
020064 001403 020066 14S:  INC     R4            ;; GO TYPE THIS DIGIT
020066 005204 020070 15S:  BIS     #'0,R3        ;; COUNT BY 1
020070 052703 000060 16S:  BIS     #' ,R3        ;; BR IF MORE TO DO
020074 052703 000040 17S:  MOVVB  R3,%S        ;; BR IF DONE
020100 110337 020104 18S:  TYPE   #S            ;; INSURE LAST DIGIT ISN'T A BLANK
020104 104401 020144 19S:  DECIB  %SOCNT        ;; GO DO THE LAST DIGIT
020110 105337 020146 20S:  BGT     25S          ;; RESTORE R5
020114 003347 020146 21S:  BLT     6S            ;; RESTORE R4
020116 002402 020146 22S:  BLT     6S            ;; RESTORE R3
020120 005204 020146 23S:  INC     R4            ;; SET THE STACK FOR RETURNING
020122 000744 020146 24S:  BR      25S          ;; RESTORE R5
020124 012605 020146 25S:  MOV     (SP)+,R5      ;; RESTORE R4
020126 012604 020146 26S:  MOV     (SP)+,R4      ;; RESTORE R3
020130 012603 020146 27S:  MOV     (SP)+,R3      ;; RESTORE R3
020132 01666E 000002 000004 28S:  MOV     2(SP),4 SP    ;; SET THE STACK FOR RETURNING

```

4278 020140 012616
4279 020142 000002
4280 020144 000
4281 020146 000
4282 020148 000
4283 020150 000000

```
MOV (SP)+,(SP)
RTI
B$: .BYTE 0
SOCNT: .BYTE 0
SOFILL: .BYTE 0
SOMODE: .WORD 0
:: RETURN
:: STORAGE FOR ASCII DIGIT
:: TERMINATOR FOR TYPE ROUTINE
:: OCTAL DIGIT COUNTER
:: ZERO FILL SWITCH
:: NUMBER OF DIGITS TO TYPE
```

.SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

```
:: *****
:: THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
:: SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
:: NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
:: BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
:: REPLACED WITH SPACES.
```

```
:: CALL:
* MOV NUM,-(SP) :: PUT THE BINARY NUMBER ON THE STACK
* TYPDS :: GO TO THE ROUTINE
```

```
$TYPDS:
MOV R0,-(SP) :: PUSH R0 ON STACK
MOV R1,-(SP) :: PUSH R1 ON STACK
MOV R2,-(SP) :: PUSH R2 ON STACK
MOV R3,-(SP) :: PUSH R3 ON STACK
MOV R5,-(SP) :: PUSH R5 ON STACK
MOV #20200,-(SP) :: SET BLANK SWITCH AND SIGN
MOV 20(SP),R5 :: GET THE INPUT NUMBER
BPL R5 :: BR IF INPUT IS POS.
NEG R5 :: MAKE THE BINARY NUMBER POS.
MOVB #-1(SP) :: MAKE THE ASCII NUMBER NEG.
CLR R0 :: ZERO THE CONSTANTS INDEX
MOV #0BLK,R3 :: SETUP THE OUTPUT POINTER
MOVB #1(R3)+ :: SET THE FIRST CHARACTER TO A BLANK
CLR R2 :: CLEAR THE BCD NUMBER
MOV $0TB(R0),R1 :: GET THE CONSTANT
SUB R1,R5 :: FORM THIS BCD DIGIT
BLT R5 :: BR IF DONE
INC R2 :: INCREASE THE BCD DIGIT BY 1
BR 3$
4$: ADD R1,R5 :: ADD BACK THE CONSTANT
TST R2 :: CHECK IF BCD DIGIT=0
BNE 5$ :: FALL THROUGH IF 0
TSTB (SP) :: STILL DOING LEADING 0'S?
BMI 7$ :: BR IF YES
5$: ASLB (SP) :: MSD?
BCC 6$ :: BR IF NO
MOVB 1(SP),-1(R3) :: YES--SET THE SIGN
6$: BIS #'C,R2 :: MAKE THE BCD DIGIT ASCII
7$: BIS #' ',R2 :: MAKE IT A SPACE IF NOT ALREADY A DIGIT
MOV R2,R3+ :: PUT THIS CHARACTER IN THE OUTPUT BUFFER
TST R0+ :: JUST INCREMENTING
CMP R0,#10 :: CHECK THE TABLE INDEX
BLT R5 :: GO DO THE NEXT DIGIT
BGT R5 :: GO TO EXIT
MOV R5,R2 :: GET THE LSD
```

4284 020152 010046
4285 020154 010146
4286 020156 010246
4287 020158 010346
4288 020160 010446
4289 020162 010546
4290 020164 012746 020200
4291 020170 016605 00002C
4292 020174 100004
4293 020176 005405
4294 020200 112766 000055 000001
4295 020206 005000 1\$:
4296 020210 012703 020366
4297 020214 112723 000040
4298 020220 005002 2\$:
4299 020222 016001 020356
4300 020226 160105 3\$:
4301 020230 002402
4302 020232 005202
4303 020234 000774
4304 020236 060105 4\$:
4305 020240 005702
4306 020242 001002
4307 020244 105716
4308 020246 100407
4309 020250 106316 5\$:
4310 020252 103003
4311 020254 116663 000001 177777
4312 020262 052702 000060 6\$:
4313 020266 052702 000040 7\$:
4314 020272 110223
4315 020274 005720
4316 020276 020027 000010
4317 020302 002746
4318 020304 003002
4319 020306 010502

```

1334 020310 000764
1335 020312 105726
1336 020314 100003
1337 020316 116663 177777 177776
1338 020318 105013
1339 020320 012603
1340 020322 012603
1341 020324 012601
1342 020326 012600
1343 020328 104401 020366 000004
1344 020330 016666 000002
1345 020332 012616
1346 020334 000002
1347 020336 023420
1348 020338 001750
1349 020340 000144
1350 020342 000012
1351 020344 000004
1352
1353
1354
1355
1356
1357
1358 020376 000000
1359 020400 000000
1360 020402 000000
1361 020404 000012
1362 020416
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372 020416 005037 020376
1373 020422 012737 020404 020400
1374 020430 013737 020400 020402
1375 020436 012737 020466 000060
1376 020444 012737 000200 000062
1377 020452 005777 160470
1378 020456 012777 00C100 160460
1379 020464 000207
1380
1381
1382
1383
1384
1385
1386
1387
1388 020466 117746 160454
1389 020472 042716 177500

```

```

BR 6$ :: GO CHANGE TO ASCII
TSTB (SP)+ :: WAS THE LSD THE FIRST NON-ZERO
BPL 9$ :: BR IF NO
MOVB -1(SP),-2(R3) :: YES--SET THE SIGN FOR TYPING
CLRB (R3) :: SET THE TERMINATOR
MOV (SP)+,R5 :: POP STACK INTO R5
MOV (SP)+,R3 :: POP STACK INTO R3
MOV (SP)+,R2 :: POP STACK INTO R2
MOV (SP)+,R1 :: POP STACK INTO R1
MOV (SP)+,R0 :: POP STACK INTO R0
TYPE $DBLK :: NOW TYPE THE NUMBER
MOV 2(SP),4(SP) :: ADJUST THE STACK
MOV (SP)+,(SP)
RTI :: RETURN TO USER

$DTBL: 1000.
1000.
100.
10.
$DBLK: .BLKW 4

.SBTTL TTY INPUT ROUTINE
*****
ENABL LSB
$TKCNT: .WORD 0 :: NUMBER OF ITEMS IN QUEUE
$TKQIN: .WORD 0 :: INPUT POINTER
$TKQOUT: .WORD 0 :: OUTPUT POINTER
$TKQSRV: .BLKB 10. :: TTY KEYBOARD QUEUE
$TKQEND=.

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

*TK SERVICE ROUTINE
*THIS ROUTINE WILL SERVICE THE TTY KEYBOARD INTERRUPT
*BY READING THE CHARACTER FROM THE INPUT BUFFER AND PUTTING
*IT IN THE QUEUE.
*IF THE CHARACTER IS A "CONTROL-C" (1C) $TKINT IS CALLED AND
*UPON RETURN EXIT IS MADE TO THE "CONTROL-C" RESTART ADDRESS START2
$TKSRV: MOVB $TKB,-(SP) :: PICKUP THE CHARACTER
BIC #1C177,SP :: STRIP THE JUNK

```

```

4390 020476 021627 000003      CMP      (SP),#3      :: IS IT A CONTROL C?
4391 020502 001007      BNE      1$          :: BRANCH IF NO
4392 020504 104401 021604      TYPE    $CNTLC      :: TYPE A CONTROL-C (IC
4393 020510 004737 020416      JSR     PC,STKINT   :: INIT THE KEYBOARD
4394 020514 005726      TST     (SP)+       :: CLEAN UP STACK
4395 020516 000137 003676      JMP     START2      :: CONTROL C RESTART
4396 020520 021627 000007      CMP     (SP),#7     :: IS IT A CONTROL G?
4397 020526 001004      BNE      2$          :: BRANCH IF NO
4398 020530 022737 000176 001140      CMP     $SWREG,SWR  :: IS SOFT-SWR SELECTED?
4399 020536 001500      BEQ     6$          :: GO TO SWR CHANGE
4400
4401 020540      2$:
4402 020540 022737 000012 020376      CMP     #10,$STKCNT :: IS THE QUEUE FULL?
4403 020546 001004      BNE      3$          :: BRANCH IF NO
4404 020550 104401 001166      TYPE    $BELL       :: RING THE TTY BELL
4405 020554 005726      TST     (SP)+       :: CLEAN CHARACTER OFF OF STACK
4406 020556 000451      BR      5$          :: EXIT
4407 020560 021627 000023      CMP     (SP),#23    :: IS IT A CONTROL-S?
4408 020564 001021      BNE      32$         :: BRANCH IF NO
4409 020566 005077 160352      CLR     $STKS       :: DISABLE TTY KEYBOARD INTERRUPTS
4410 020572 005726      TST     (SP)+       :: CLEAN CHAR OFF STACK
4411 020574 105777 160344      TSTB   $STKS       :: WAIT FOR A CHAR
4412 020600 100375      BPL     31$         :: LOOP UNTIL ITS THERE
4413 020602 117746 160340      MOVB   $STKB,-(SP) :: GET THE CHARACTER
4414 020606 042716 177600      BIC    #177,(SP)   :: MAKE IT 7-BIT ASCII
4415 020612 022627 000021      CMP     (SP)+,#21   :: IS IT A CONTROL-Q?
4416 020616 001366      BNE      31$         :: BRANCH IF NO
4417 020620 012777 000100 160316      MOV     #100,$STKS :: REENABLE TTY KEYBOARD INTERRUPTS
4418 020626 000002      RTI
4419 020630 005237 020376      32$: INC     $STKCNT  :: COUNT THIS CHARACTER
4420 020634 021627 000140      CMP     (SP),#140  :: IS IT UPPER CASE?
4421 020640 002405      BLT     4$          :: BRANCH IF YES
4422 020642 021627 000175      CMP     (SP),#175  :: IS IT A SPECIAL CHAR?
4423 020646 003002      BGT     4$          :: BRANCH IF YES
4424 020650 042716 000040      BIC    #40,(SP)   :: MAKE IT UPPER CASE
4425 020654 112677 177520      MOVB   (SP)+,$STKQIN :: AND PUT IT IN QUEUE
4426 020660 005237 020400      INC     $STKQIN   :: UPDATE THE POINTER
4427 020664 023727 020400 020416      CMP     $STKQIN,$STKQEND :: GO OFF THE END?
4428 020672 001003      BNE      5$          :: BRANCH IF NO
4429 020674 012737 020404 020400      MOV     #STKQSRST,$STKQIN :: RESET THE POINTER
4430 020702 000002      5$: RTI
4431
4432 *****
4433 *SOFTWARE SWITCH REGISTER CHANGE ROUTINE
4434 *ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
4435 *SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP
4436 *CALL WHEN OPERATING IN TTY INTERRUPT MODE.
4437 020704 022737 000176 001140 $CKSWR: CMP     #SWREG,SWR  :: IS THE SOFT-SWR SELECTED
4438 020712 001124      BNE      15$         :: EXIT IF NOT
4439 020714 105777 160224      TSTB   $STKS       :: IS A CHAR WAITING?
4440 020720 100121      BPL     15$         :: IF NOT, EXIT
4441 020722 117746 160220      MOVB   $STKB,-(SP) :: YES
4442 020726 042716 177600      BIC    #177,(SP)   :: MAKE IT 7-BIT ASCII
4443 020732 021627 000007      CMP     (SP),#7     :: IS IT A CONTROL-G?
4444 020736 001300      BNE      2$          :: IF NOT, PUT IT IN THE TTY QUEUE
4445      2$: AND EXIT

```

```

*****
*CONTROL IS PASSED TO THIS POINT FROM EITHER THE TTY INTERRUPT SERVICE
*ROUTINE OR FROM THE SOFTWARE SWITCH REGISTER TRAP CALL AS A RESULT OF A
*CONTROL-G BEING TYPED, AND THE SOFTWARE SWITCH REGISTER BEING SELECTED.
*****
4446 020740 123727 001134 000001 6S:  CMPB  $AUTOB,#1  :: ARE WE RUNNING IN AUTO-MODE?
4447 020746 001674  BEQ  2S  :: BRANCH IF YES
4448 020750 005726  TST  (SP)+  :: CLEAR CONTROL-G OFF STACK
4449 020752 004737 020416  JSR  PC,$TKINT  :: FLUSH THE TTY INPUT QUEUE
4450 020756 005077 160162  CLR  $TKS  :: DISABLE TTY KEYBOARD INTERRUPTS
4451 020762 112737 000001 001135  MOVB  #1,$INTAG  :: SET INTERRUPT MODE INDICATOR
4452 020770 104401 021616  SGTSWR: TYPE  $SCNTLG  :: ECHO THE CONTROL-G (1G)
4453 020774 104401 021623  TYPE  $MSWR  :: TYPE CURRENT CONTENTS
4454 021000 013746 000176  MOV  SWREG,-(SP)  :: SAVE SWREG FOR TYPEOUT
4455 021004 104402  TYPEOC  :: GO TYPE--OCTAL ASCII(ALL DIGITS)
4456 021006 104401 021634  TYPE  $MNEW  :: PROMPT FOR NEW SWR
4457 021012 005046 19S:  CLR  -(SP)  :: CLEAR COUNTER
4458 021014 005046  CLR  -(SP)  :: THE NEW SWR
4459 021016 105777 160122  TS:  TSTB  $TKS  :: CHAR THERE?
4460 021022 100375  BPL  7S  :: IF NOT TRY AGAIN
4461 021024 117746 160116  MOVB  $TKB,-(SP)  :: PICK UP CHAR
4462 021030 042716 177600  BIC  #1C177,(SP)  :: MAKE IT 7-BIT ASCII
4463 021034 021627 000003  CMP  (SP),#3  :: IS IT A CONTROL-C?
4464 021040 001015  BNE  9S  :: BRANCH IF NOT
4465 021042 104401 021604  TYPE  $SCNTLC  :: YES, ECHO CONTROL-C (1C)
4466 021046 062706 000006  ADD  #6,SP  :: CLEAN UP STACK
4467 021052 123727 001135 000001  CMPB  $INTAG,#1  :: REENABLE TTY KEYBOARD INTERRUPTS?
4468 021050 001003  BNE  8S  :: BRANCH IF NO
4469 021062 012777 000100 160054  MOV  #100,$TKS  :: ALLOW TTY KEYBOARD INTERRUPTS
4470 021070 000137 003676  8S:  JMP  START2  :: CONTROL-C RESTART
4471 021074 021627 000025  9S:  CMP  (SP),#25  :: IS IT A CONTROL-U?
4472 021100 001005  BNE  10S  :: BRANCH IF NOT
4473 021102 104401 021611  TYPE  $SCNTLU  :: YES, ECHO CONTROL-U (1U)
4474 021106 062706 000006  20S:  ADD  #6,SP  :: IGNORE PREVIOUS INPUT
4475 021112 000737  BR  19S  :: LET'S TRY IT AGAIN
4476 021114 021627 000015  10S:  CMP  (SP),#15  :: IS IT A <CR>?
4477 021120 001022  BNE  16S  :: BRANCH IF NO
4478 021122 005766 000004  TST  4(SP)  :: YES, IS IT THE FIRST CHAR?
4479 021126 001403  BEQ  11S  :: BRANCH IF YES
4480 021130 016677 000002 160002  MOV  2(SP), $SWR  :: SAVE NEW SWR
4481 021136 062706 000006  11S:  ADD  #6,SP  :: CLEAR UP STACK
4482 021142 104401 001173  14S:  TYPE  $SCRLF  :: ECHO <CR> AND <LF>
4483 021146 123727 001135 000001  CMPB  $INTAG,#1  :: RE-ENABLE TTY KBD INTERRUPTS?
4484 021154 001003  BNE  15S  :: BRANCH IF NOT
4485 021156 012777 000100 157760  MOV  #100,$TKS  :: RE-ENABLE TTY KBD INTERRUPTS
4486 021164 000002  15S:  RTI  :: RETURN
4487 021166 004737 017654  16S:  JSR  PC,$TYPEC  :: ECHO CHAR
4488 021172 021627 000060  CMP  (SP),#60  :: CHAR < 0?
4489 021176 002420  BLT  18S  :: BRANCH IF YES

```

TTY INPUT ROUTINE

```

4502 021200 021627 000067      CMP      SP), #67      ;; CHAR , ??
4503 021204 003015      BGT      18$          ;; BRANCH IF YES
4504 021206 042726 000060      BIC      #60, (SP)+  ;; STRIP-OFF ASCII
4505 021212 005766 000002      TST      2(SP)       ;; IS THIS THE FIRST CHAR
4506 021216 001403      BEQ      17$          ;; BRANCH IF YES
4507 021220 006316      ASL      (SP)        ;; NO, SHIFT PRESENT
4508 021222 006316      ASL      (SP)        ;; CHAR OVER TO MAKE
4509 021224 006316      ASL      (SP)        ;; ROOM FOR NEW ONE.
4510 021226 005266 000002      17$: INC      2(SP)    ;; KEEP COUNT OF CHAR
4511 021230 056616 177776      BR       -2(SP), (SP) ;; SET IN NEW CHAR
4512 021236 000667      BR       7$          ;; GET THE NEXT ONE
4513 021240 104401 001172      18$: TYPE   $QUES    ;; TYPE ? <CR> <LF>
4514 021244 000720      BR       20$        ;; SIMULATE CONTROL-L
4515      .DSABL  LSB
4516
4517
4518      ;; *****
4519      ;; THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
4520      ;; *CALL:
4521      ;; *      RDCHR          ;; GET A CHARACTER FROM THE QUEUE
4522      ;; *      RETURN HERE   ;; CHARACTER IS ON THE STACK
4523      ;; *                  ;; WITH PARITY BIT STRIPPED OFF
4524
4525
4526 021246 011646      $RDCHR: MOV      (SP), -(SP) ;; PUSH DOWN THE PC AND
4527 021250 016666 000004 000002      MOV      4(SP), 2(SP) ;; THE PS
4528 021256 005066 000004      CLR      4(SP)        ;; GET READY FOR A CHARACTER
4529 021262 005046      CLR      -(SP)       ;; PUT NEW PS ON STACK
4530 021264 012746 021272      MOV      #64$, -(SP) ;; PUT NEW PC ON STACK
4531 021270 000002      RTI                ;; POP NEW PC AND PS
4532 021272
4533 021272 005737 020376      64$: TST      $TKCNT    ;; WAIT ON A CHARACTER
4534 021276 001775      1$: BEQ      1$
4535 021300 005337 020376      DEC      $TKCNT      ;; DECREMENT THE COUNTER
4536 021304 117766 177072 000004      MOVVB   $TKQOUT, 4(SP) ;; GET ONE CHARACTER
4537 021312 005237 020402      INC      $TKQOUT     ;; UPDATE THE POINTER
4538 021316 023727 020402 020416      CMP      $TKQOUT, # $TKQEND ;; DID IT GO OFF OF THE END?
4539 021324 001003      BNE     2$          ;; BRANCH IF NO
4540 021326 012737 020404 020402      MOV      # $TKQSRT, $TKQOUT ;; RESET THE POINTER
4541 021334 000002      2$: RTI          ;; RETURN
4542
4543      ;; *****
4544      ;; THIS ROUTINE WILL INPUT A STRING FROM THE TTY
4545      ;; *CALL:
4546      ;; *      RDLIN        ;; INPUT A STRING FROM THE TTY
4547      ;; *      RETURN HERE  ;; ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
4548      ;; *                  ;; TERMINATOR WILL BE A BYTE OF ALL 0'S
4549
4550 021336 010346      $RDLIN: MOV      R3, -(SP) ;; SAVE R3
4551 021340 005046      CLR      -(SP)       ;; CLEAR THE RUBOUT KEY
4552 021342 012703 021572      1$: MOV      # $TTYIN, R3 ;; GET ADDRESS
4553 021346 022703 021604      2$: CMP      # $TTYIN+10., R3 ;; BUFFER FULL?
4554 021352 101456      BLOS    4$          ;; BR IF YES
4555 021354 104410      RDCHR   ;; GO READ ONE CHARACTER FROM THE TTY
4556 021356 112613      MOVVB   (SP)+, (R3)  ;; GET CHARACTER
4557 021360 122713 000177      10$: CMPB   #177, (R3) ;; IS IT A RUBOUT
4558 021364 001022      BNE     5$          ;; BR IF NO

```

TTY INPUT ROUTINE

```

4558 021366 005716          TST      (SP)          ;; IS THIS THE FIRST RUBOUT?
4559 021370 001007          BNE     6$            ;; BR IF NO
4560 021372 112737 000134 021570  MOVB    #' \,9$      ;; TYPE A BACK SLASH
4561 021400 104401 021570      TYPE    9$
4562 021404 012716 177777      MOV     4-1,(SP)     ;; SET THE RUBOUT KEY
4563 021410 005303          DEC     R3           ;; BACKUP BY ONE
4564 021412 020327 021572      CMP     R3,#$TTYIN  ;; STACK EMPTY?
4565 021416 103434          BLO    4$           ;; BR IF YES
4566 021420 111337 021570      MOVB   (R3),9$      ;; SETUP TO TYPEOUT THE DELETED CHAR.
4567 021424 104401 021570      TYPE   9$          ;; GO TYPE
4568 021430 000746          BR     2$           ;; GO READ ANOTHER CHAR.
4569 021432 005716          TST    (SP)         ;; RUBOUT KEY SET?
4570 021434 001406          BEQ    7$          ;; BR IF NO
4571 021436 112737 000134 021570  MOVB    #' \,9$      ;; TYPE A BACK SLASH
4572 021444 104401 021570      TYPE    9$
4573 021450 095016          CLR    (SP)         ;; CLEAR THE RUBOUT KEY
4574 021452 122713 000025      CMPB   #25,(R3)    ;; IS CHARACTER A CTRL U?
4575 021456 001003          BNE    8$          ;; BR IF NO
4576 021460 104401 021611      TYPE   $CNTLU      ;; TYPE A CONTROL "U"
4577 021464 000726          BR     1$          ;; GO START OVER
4578 021466 122713 000022      CMPB   #22,(R3)    ;; IS CHARACTER A "r"?
4579 021472 001011          BNE    3$          ;; BRANCH IF NO
4580 021474 105013          CLRB   (R3)        ;; CLEAR THE CHARACTER
4581 021476 104401 001173      TYPE   $SCRLF      ;; TYPE A "CR" & "LF"
4582 021502 104401 021572      TYPE   $TTYIN      ;; TYPE THE INPUT STRING
4583 021506 000717          BR     2$          ;; GO PICKUP ANOTHER CHARACTER
4584 021510 104401 001172      TYPE   $QUES       ;; TYPE A '?'
4585 021514 000712          BR     1$          ;; CLEAR THE BUFFER AND LOOP
4586 021516 111337 021570      MOVB   (R3),9$     ;; ECHO THE CHARACTER
4587 021522 104401 021570      TYPE   9$
4588 021526 122723 000015      CMPB   #15,(R3)+  ;; CHECK FOR RETURN
4589 021532 001305          BNE    2$          ;; LOOP IF NOT RETURN
4590 021534 105063 177777      CLRB   -1(R3)     ;; CLEAR RETURN (THE 15)
4591 021540 104401 001174      TYPE   $LF         ;; TYPE A LINE FEED
4592 021544 005726          TST   (SP)+        ;; CLEAN RUBOUT KEY FROM THE STACK
4593 021546 012603          MOV    (SP)+,R3    ;; RESTORE R3
4594 021550 011646          MOV    (SP)-,(SP)  ;; ADJUST THE STACK AND PUT ADDRESS OF THE
4595 021552 016666 000004 000002  MOV     4(SP),2(SP) ;; FIRST ASCII CHARACTER ON IT
4596 021560 012766 021572 000004  MOV     #$TTYIN,4(SP)
4597 021566 000002          RTI
4598 021570 000          9$: .BYTE 0          ;; RETURN
4599 021571 000          .BYTE 0          ;; STORAGE FOR ASCII CHAR. TO TYPE
4600 021572 000012          $TTYIN: .BLKB 10  ;; TERMINATOR
4601 021604 041536 005015 000  $CNTLC: .ASCIZ /?C<15><12>  ;; RESERVE 10. BYTES FOR TTY INPUT
4602 021611 136 006525 000012  $CNTLU: .ASCIZ /?U<15><12>  ;; CONTROL "C"
4603 021616 043536 005015 000  $CNTLG: .ASCIZ /?G<15><12>  ;; CONTROL "U"
4604 021623 015 051412 051127  $MSWR: .ASCIZ <15><12>/SWR = /  ;; CONTROL "G"
4605 021630 036440 000040          $MNEW: .ASCIZ / NEW = /
4606 021634 020040 042516 020127
4607 021642 020075 000
4608 021646          .EVEN
4609
4610          .SBTTL SCOPE HANDLER ROUTINE
4611
4612          ;; *****
4613          ;; *THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT

```

```

4614
4615
4616
4617
4618
4619
4620
4621
4622
4623 021646
4624 021646 104407
4625 021650 032777 040000 157262
4626 021656 001101
4627
4628 021660 0004.6
4629
4630 021662 013746 000004
4631 021666 012737 021706 000004
4632 021674 005737 177060
4633 021700 012637 000004
4634 021704 000453
4635 021706 022626
4636 021710 012637 000004
4637 021714 000413
4638 021716
4639 021716 105737 001103
4640 021722 001421
4641 021724 123737 001115 001103
4642 021732 101015
4643 021734 032777 001000 157176
4644 021742 001404
4645 021744 013737 001110 001106
4646 021752 000443
4647 021754 105037 001103
4648 021760 005037 001162
4649 021764 000415
4650 021766 032777 004000 157144
4651 021774 001011
4652 021776 005737 001100
4653 022002 001406
4654 022004 005237 001104
4655 022010 023737 001162 001104
4656 022016 002021
4657 022020 012737 000001 001104
4658 022026 013737 022076 001162
4659 022034 105237 001102
4660 022040 011637 001106
4661 022044 011637 001110
4662 022050 005037 001164
4663 022054 112737 000001 001115
4664 022062 013777 001102 157052
4665 022070 013716 001106
4666 022074 000002
4667 022076 000001
4668
4669

```

```

;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY 7:0.)
;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY(15:08)
;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
;*SW14=1 LOOP ON TEST
;*SW11=1 INHIBIT ITERATIONS
;*SW09=1 LOOP ON ERROR
;*CALL
* SCOPE ;;SCOPE=IOT

$SCOPE:
CKSWR
1$: BIT #BIT14,$SWR ;;TEST FOR CHANGE IN SOFT-SWR
BNE $OVER ;;LOOP ON PRESENT TEST?
;;YES IF SW14=1
*****START OF CODE FOR THE XOR TESTER*****
$XSTR: BR 6$ ;;IF RUNNING ON THE "XOR" TESTER CHANGE
;;THIS INSTRUCTION TO A "NOP" (NOP=240)
MOV $ERRVEC -(SP) ;;SAVE THE CONTENTS OF THE ERROR VECTOR
MOV $S $ERRVEC ;;SET FOR TIMEOUT
TST $177060 ;;TIME OUT ON XOR?
BR ($SP)+,$ERRVEC ;;RESTORE THE ERROR VECTOR
$SVLAD BR ;;GO TO THE NEXT TEST
5$: CMP ($SP)+,($SP)+ ;;CLEAR THE STACK AFTER A TIME OUT
MOV ($SP)+,$ERRVEC ;;RESTORE THE ERROR VECTOR
BR 7$ ;;LOOP ON THE PRESENT TEST
6$; *****END OF CODE FOR THE XOR TESTER*****
2$: TSTB $ERFLG ;;HAS AN ERROR OCCURRED?
BEQ 3$ ;;BR IF NO
CMPB $ERMAX,$ERFLG ;;MAX. ERRORS FOR THIS TEST OCCURRED?
BHI 3$ ;;BR IF NO
BIT #BIT09,$SWR ;;LOOP ON ERROR?
BEQ 4$ ;;BR IF NO
7$: MOV $LPERR,$LPADR ;;SET LOOP ADDRESS TO LAST SCOPE
BR $OVER
4$: CLRB $ERFLG ;;ZERO THE ERROR FLAG
CLR $TIMES ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
BR 1$ ;;ESCAPE TO THE NEXT TEST
3$: BIT #BIT11,$SWR ;;INHIBIT ITERATIONS?
BNE 1$ ;;BR IF YES
TST $PASS ;;IF FIRST PASS OF PROGRAM
BEQ 1$ ;;INHIBIT ITERATIONS
INC $ICNT ;;INCREMENT ITERATION COUNT
CMP $TIMES,$ICNT ;;CHECK THE NUMBER OF ITERATIONS MADE
BGE $OVER ;;BR IF MORE ITERATION REQUIRED
1$: MOV #1,$ICNT ;;REINITIALIZE THE ITERATION COUNTER
MOV $MXCNT,$TIMES ;;SET NUMBER OF ITERATIONS TO DO
$SVLAD: INCB $TSTNM ;;COUNT TEST NUMBERS
MOV (SP),$LPADR ;;SAVE SCOPE LOOP ADDRESS
MOV (SP),$LPERR ;;SAVE ERROR LOOP ADDRESS
CLR $ESCAPE ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
MOVB #1,$ERMAX ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
$OVER: MOV $TSTNM,$DISPLAY ;;DISPLAY TEST NUMBER
MOV $LPADR,(SP) ;;FUDGE RETURN ADDRESS
RTI ;;FIXES PS
$MXCNT: 1 ;;MAX. NUMBER OF ITERATIONS

.SBTTL SAVE AND RESTORE PO-R5 ROUTINES

```


4670
4671
4672
4673
4674
4675
4676
4677
4678
4679
4680
4681
4682
4683
4684
4685
4686
4687
4688
4689
4690
4691
4692
4693
4694
4695
4696
4697
4698
4699
4700
4701
4702
4703
4704
4705
4706
4707
4708
4709
4710
4711
4712
4713
4714
4715
4716
4717
4718
4719
4720
4721
4722
4723
4724
4725

022100
022100 010046
022102 010146
022104 010246
022106 010346
022110 010446
022112 010546
022114 016646 000022
022120 016646 000022
022124 016646 000022
022130 016646 000022
022134 000002

022136
022136 012666 000022
022142 012666 000022
022146 012666 000022
022152 012666 000022
022156 012605
022160 012604
022162 012603
022164 012602
022166 012601
022170 012600
022172 000002

```
*****  
:SAVE RO-R5  
:CALL:  
: SAVREG  
:UPON RETURN FROM $$SAVREG THE STACK WILL LOOK LIKE:  
:  
:TOP---(+16)  
: +2---(+18)  
: +4---R5  
: +6---R4  
: +8---R3  
: +10---R2  
: +12---R1  
: +14---R0
```

```
$$SAVREG:  
MOV R0, -(SP) :PUSH R0 ON STACK  
MOV R1, -(SP) :PUSH R1 ON STACK  
MOV R2, -(SP) :PUSH R2 ON STACK  
MOV R3, -(SP) :PUSH R3 ON STACK  
MOV R4, -(SP) :PUSH R4 ON STACK  
MOV R5, -(SP) :PUSH R5 ON STACK  
MOV R2(SP), -(SP) :SAVE PS OF MAIN FLOW  
MOV R2(SP), -(SP) :SAVE PC OF MAIN FLOW  
MOV R2(SP), -(SP) :SAVE PS OF CALL  
MOV R2(SP), -(SP) :SAVE PC OF CALL  
RTI
```

```
:*RESTORE RO-R5  
:CALL:  
: RESREG  
$$RESREG:  
MOV (SP)+, R2(SP) :RESTORE PC OF CALL  
MOV (SP)+, R2(SP) :RESTORE PS OF CALL  
MOV (SP)+, R2(SP) :RESTORE PC OF MAIN FLOW  
MOV (SP)+, R2(SP) :RESTORE PS OF MAIN FLOW  
MOV (SP)+, R5 :POP STACK INTO R5  
MOV (SP)+, R4 :POP STACK INTO R4  
MOV (SP)+, R3 :POP STACK INTO R3  
MOV (SP)+, R2 :POP STACK INTO R2  
MOV (SP)+, R1 :POP STACK INTO R1  
MOV (SP)+, R0 :POP STACK INTO R0  
RTI
```

.SBTTL ROUTINE TO SIZE MEMORY

```
*****  
:CALL:  
: JSR PC, $SIZE  
: RETURN  
:*$LSTAD WILL CONTAIN:  
: WITH KT11 OPTION -- LAST VIRTUAL ADDRESS OF THE LAST BANK  
: WITHOUT KT11 OPTION -- LAST ABSOLUTE ADDRESS OF AVAILABLE MEMORY  
:*$LSTBK WILL CONTAIN THE LAST BANK AS A SAF  
:*$KT11 IS THE MEMORY MANAGEMENT KEY
```

```

4726 ;#BIT07 = 0 DON'T USE MEMORY MANAGEMENT
4727 ;# MUST BE SETUP BEFORE THE CALL
4728 ;#BIT15 = 0 DON'T HAVE MEMORY MANAGEMENT OPTION
4729 ;# DETERMINED BY ROUTINE
4730
4731 $SIZE: MOV R0, -(SP) ;: SAVE R0 ON THE STACK
4732 MOV R1, -(SP) ;: SAVE R1 ON THE STACK
4733 MOV R2, -(SP) ;: SAVE R2 ON THE STACK
4734 MOV R3, -(SP) ;: SAVE R3 ON THE STACK
4735 MOV @#ERRVEC, -(SP) ;: SAVE PRESENT ERROR VECTOR PS & PC
4736 MOV @#ERRVEC+2, -(SP)
4737 MOV SP, R0 ;: SAVE THE STACK POINTER
4738 ;: SET THE ERRVEC PS TO THE PRESENT PS
4739 TRAP ;: PUSH OLD PSW AND PC ON STACK
4740 MOV (SP)+, @#ERRVEC+2 ;: SAVE THE PSW IN @#ERRVEC+2
4741 MOV #3776, R1 ;: SETUP ADDRESS
4742 TSTB (PC)+ ;: USE MEMORY MANAGEMENT?
4743 $KT11: WORD 200 ;: SET TO USE MEMORY MANAGEMENT
4744 BPL $SCORE ;: BR IF NO
4745 MOV @#KTNEX, @#ERRVEC ;: SET FOR TIMEOUT
4746 TST @#SR0 ;: KT11 ARE YOU THERE?
4747 BIS #100000, $KT11 ;: YES--SET KT11 KEY
4748 CLR -(SP) ;: INITIALIZE FOR "PAR" LOADING
4749 MOV @#KIPAR0, R2 ;: ADDRESS OF FIRST "PAR"
4750 MOV @#D8, R3 ;: LOAD EIGHT "PAR."S AND EIGHT "PDR."S
4751 MOV @77406, -40(R2) ;: PDR = 4K UP, READ/WRITE
4752 MOV (SP), (R2)+ ;: LOAD "PAR"
4753 ADD #200, (SP) ;: UPDATE FOR NEXT "PAR"
4754 SOB R3, 1$ ;: LOOP UNTIL ALL EIGHT ARE LOADED
4755 MOV @177600, -(R2) ;: SETUP KIPAR7 FOR I/O
4756 CLR -(R2) ;: SETUP KIPAR6 FOR TESTING
4757 MOV @25, @#ERRVEC ;: CATCH TIMEOUT IF NO SR3
4758 MOV @20, @#SR3 ;: ENABLE 22 BIT MODE
4759 BR 3$ ;: THIS PDP-11 HAS A SR3 REGISTER
4760 CMP (SP)+, (SP)+ ;: CLEAN OFF THE STACK--NO SR3
4761 INC @#SR0 ;: TURN ON MEMORY MANAGEMENT
4762 MOV @#KTOUT, @#ERRVEC ;: SET FOR TIME OUT
4763 TST @#143776 ;: TRAP ON NON-EX-MEM
4764 ADD #40, (R2) ;: MAKE A 1K STEP
4765 CMP @#KIPAR7, (R2) ;: LAST ONE?
4766 BHI 4$ ;: NO--TRY IT
4767 $KTOUT: MOV (R2), R2 ;: GET LAST BANK+1
4768 CLR @#SR0 ;: TURN OFF MEMORY MANAGEMENT
4769 BR $SIZE
4770 $KTNEX: BIC #100000, $KT11 ;: KT11 NON-EXISTENT
4771 $SCORE: MOV @#SCROUT, @#ERRVEC ;: SET FOR TIMEOUT
4772 CLR R2 ;: SET UP BANK
4773 ADD #4000, R1 ;: INCREMENT BY 1K
4774 ADD #40, R2 ;: 1K STEP
4775 TST (R1) ;: TRAP ON TIME OUT
4776 CMP @177776, R1 ;: LAST ONE
4777 BNE 1$ ;: NO--TRY AGAIN
4778 $SCROUT: SUB #4000, R1 ;: DROP BACK
4779 $SIZEX: SUB #40, R2 ;: RESTORE THE STACK
4780 MOV R0, SP ;: RESTORE ERROR VECTOR
4781 MOV (SP)+, @#ERRVEC+2

```

4780
4781
4782
4783
4784
4785
4786
4787
4788
4789
4790
4791
4792
4793
4794
4795
4796
4797
4798
4799
4800
4801
4802
4803
4804
4805
4806
4807
4808
4809
4810
4811
4812
4813
4814
4815
4816
4817
4818
4819
4820
4821
4822
4823
4824
4825
4826
4827
4828
4829
4830
4831
4832
4833
4834
4835
4836
4837

```

012637 000004      MOV      (SP)+,2,BERRVEC
010137 022476      MOV      R1,$LSTAD      ;; LAST ADDRESS
010237 022500      MOV      R1,$LSTBK      ;; LAST BANK
012603      MOV      (SP)+,R3      ;; RESTORE R3
012602      MOV      (SP)+,R2      ;; RESTORE R2
012601      MOV      (SP)+,R1      ;; RESTORE R1
012600      MOV      (SP)+,R0      ;; RESTORE R0
000207      RTS
000000      PC
000000
000000
000000

```

```

$LSTAD: .WORD 0      ;; CONTAINS THE LAST ADDRESS
$LSTBK: .WORD 0      ;; CONTAINS THE LAST BANK

```

.SBTTL TRAP DECODER

```

*****
* THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
* AND USE IT 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
        TST      -(RO)      ;; BACKUP BY 2
        MOVB     (RO),RO      ;; GET RIGHT BYTE OF TRAP
        ASL      RC      ;; POSITION FOR INDEXING
        MOV      $TRAPAD(RO),RO  ;; INDEX TO TABLE
        RTS      RC      ;; GO TO ROUTINE

```

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

```

$TRAP2: MOV      (SP)-,(SP)      ;; MOVE 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
: -----
$TRAPAD: .WORD      $TRAP2      TRAP+1(104401)  TTY TYPEOUT ROUTINE
        $TYPE      ;; CALL=TYPE      TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
        $TYPOC     ;; CALL=TYPOC     TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
        $TYPOS     ;; CALL=TYPOS     TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
        $TYPON     ;; CALL=TYPON     TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
        $TYPDS     ;; CALL=TYPDS
        $GTSWR     ;; CALL=GTSWR     TRAP+6(104406)  GET SOFT-SWR SETTING
        $CKSWR     ;; CALL=CKSWR     TRAP+7(104407)  TEST FOR CHANGE IN SOFT-SWR
        $ROCHR     ;; CALL=ROCHR     TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
        $ROLIN     ;; CALL=ROLIN     TRAP+11(104411) TTY TYPEIN STRING ROUTINE
        $SAVREG    ;; CALL=SAVREG    TRAP+12(104412) SAVE RO-R5 ROUTINE
        $RESREG    ;; CALL=RESREG    TRAP+13(104413) RESTORE RO-R5 ROUTINE

```

```

4838
4839
4840
4841
4842
4843
4844
4845
4846
4847
4848
4849
4850
4851
4852
4853
4854
4855
4856
4857
4858
4859
4860
4861
4862
4863
4864
4865
4866
4867
4868
4869
4870
4871
4872
4873
4874
4875
4876
4877
4878
4879
4880
4881
4882
4883
4884
4885
4886
4887
4888
4889
4890
4891
4892
4893

```

```

: THIS ROUTINE CLEARS THE RP11 AND DETERMINES WHICH DRIVES ARE AVAILABLE.
: THE TABLE 'DRVSTA' IS LOADED TO REFLECT THE SYSTEM STATUS.

```

```

RPINIT: SAVREG          ;SAVE R0-R5
MOV        RPVEC,R1     ;VECTOR ADDRESS
CLR        (R1)+        ;SET INTERRUPT ADDRESS TO ZERO
MOV        RPPRIO,(R1)  ;RP11 PRIORITY
CLR        DRVSTYP      ;CLEAR DRIVE TYPE STORAGE
CLR        DRVSTA       ;SET DRIVE STATUS TO OFFLINE
CLR        DRVSTA+2     ;SET DRIVE STATUS TO OFFLINE
CLR        DRVSTA+4     ;SET DRIVE STATUS TO OFFLINE
CLR        DRVSTA+6     ;SET DRIVE STATUS TO OFFLINE
MOV        RPAOR,R4     ;PUT RP11 ADDRESS INTO R4
MOV        RCLERA,RPCS(R4) ;CLEAR THE RP11
MOV        RCLERA,RPCS(R4) ;CLEAR THE RP11 AGAIN
MOV        R7,R1        ;'DRVSTA' TABLE INDEX
1$: JSR      PC,DRVINT   ;CHECK THE DRIVE'S STATUS
        DEC      R1      ;DECREMENT THE TABLE INDEX
        BR      IF NOT FINISHED
RESREG     ;RESTORE R0-R5
RTS        PC          ;RETURN

```

```

: ROUTINE TO CHECK THE DRIVE'S STATUS. DRIVE NUMBER MUST BE IN R1

```

```

DRVINT: MOVB      R1,RPCS+1(R4) ;SELECT DRIVE
        BIT      R5UFU,RPDS(R4) ;SEE IF DRIVE UNSAFE
        BEQ      1$          ;BR IF NOT UNSAFE
        MOVB      R-1,DRVSTA(R1) ;SET INDICATOR TO 'UNSAFE'
        BR      3$          ;EXIT
1$: BIT      R5UOL,RPDS(R4)     ;IS DRIVE ONLINE ?
        BEQ      4$          ;BR IF NOT
        BIT      R5USI,RPDS(R4) ;SEEK INCOMPLETE SET ?
        BNE      2$          ;BR IF SET
        BIT      R5URDY,RPDS(R4) ;IS DRIVE READY ?
        BEQ      4$          ;BR IF NOT
2$: MOVB      R1,DRVSTA(R1) ;SET DRIVE INDICATOR TO 'ONLINE'
3$: BIT      R5URPO3,RPDS(R4) ;IS THE DRIVE AN RPO3 ?
        BEQ      4$          ;BR IF NOT
4$: BISB      ATABIT(R1),DRVSTYP ;SET RPO3 INDICATOR
        RTS        PC          ;RETURN

```

```

: DO THE REQUESTED COMMAND. THE RP11 ADDRESS MUST BE IN R4

```

```

RP11: SAVREG          ;SAVE R0 - R5
MOV      DPB,RPCS+1(R4) ;SELECT THE DRIVE
CLR      R1             ;CLEAR R1
MOV      DPB,R1         ;DRIVE ADDRESS
JSR      PC,CIR        ;SEE IF DRIVE IS READY
MOV      SWC,-(SP)     ;COMPLEMENT THE WORD COUNT
NEG      (SP)          ;COMPLEMENT
MOV      (SP)+,RPWC(R4) ;LOAD THE WORD COUNT REGISTER
MOV      RBUF,RPBA(R4) ;BUFFER ADDRESS
MOV      RCYL,RPCA(R4) ;CYLINDER ADDRESS
MOV      RSEC,RPDA(R4) ;SECTOR/TRACK ADDRESS
MOV      R24000,2$    ;SET UP FOR...

```

```

4894 023054 005337 023060 1$: DEC 2$ :...A DELAY OF
4895 023060 000000 2$: 0 :...AT LEAST
4896 023060 001374 BNE 1$ :...25 MILLISECONDS.
4897 023064 113764 002500 000005 MOVB DPB,RPCS+1(R4) :SELECT THE DRIVE
4898 023072 153764 002503 000005 BISB DPB+3,RPCS+1(R4) :SET 'MODE' & 'HDR' BITS
4899 023100 013746 002502 MOV DPB+2,-(SP) :COMMAND CODE
4900 023104 153716 002501 BISB DPB+1,(SP) :'MEX' BITS
4901 023110 112664 000004 MOVB (SP)+,RPCS(R4) :START THE COMMAND
4902 023114 104413 RESREG :RESTORE R0 - R5
4903 023116 000207 RTS PC :RETURN
4904 :SEE IF DRIVE IS STILL ONLINE
4905
4906
4907
4908 023120 CIR:
4909 023120 012737 023152 001164 MOV #2$,$ESCAPE :ESCAPE TO 2$ ON ERROR
4910 023122 004737 022666 JSR PC,DRVINT :CHECK ON THE DRIVE'S STATUS
4911 023124 004737 023240 JSR PC,SAVRP :STORE RP11 REGISTERS IN CASE OF ERROR
4912 023126 105761 001370 TSTB DRVSTA(R1) :IS THE DRIVE STILL ONLINE?
4913 023142 003014 BGT 3$ :BR IF IT IS
4914 023146 002401 BLT 1$ :BR IF DRIVE IS UNSAFE
4915 023150 104076 76 ERROR :DRIVE HAS GONE OFFLINE
4916 023152 032777 001000 155760 1$: ERROR 77 :DRIVE HAS BECOME UNSAFE
4917 023160 001357 2$: BIT #SW09,$SWR :LOOP ON THE ERROR?
4918 023162 156137 001360 001310 BNE CIR :BR IF LOOP
4919 023170 000137 016616 ATABIT(R1),DRVBA0 :SETUP TO DEASSIGN THE DRIVE
4920 023174 005037 001164 JMP #SEOP :GO TO THE END OF PASS ROUTINE
4921 023200 000207 CLR $ESCAPE :CLEAR THE 'ESCAPE' FLAG
4922 RTS PC :RETURN
4923
4924 :THIS ROUTINE WILL PROVIDE A STALL IN MILLISECONDS FOR A SPECIFIC
4925 :AMOUNT OF TIME. STALL VALUE IS STORED IN 'STALL'.
4926 :
4927 JSR PC,STALL
4928
4929
4930
4931 023202 013746 001220 STALL: MOV STALL,-(SP) :PICKUP THE STALL TIME
4932 023206 005046 CLR -(SP) :CLEAR TEMP. LOCATION
4933 023210 162766 000001 000002 1$: SUB #1,2(SP) :MORE STALL REQUIRED?
4934 023216 103406 3$: BLO 3$ :NO--BRANCH
4935 023220 012716 000454 MOV #300.,(SP) :STALL FOR ABOUT 1 MILLISECOND
4936 023224 005700 2$: TST R0 :NOP TO KILL TIME
4937 023226 005316 DEC (SP) :COUNT
4938 023230 001375 BNE 2$ :LOOP IF MORE COUNTS NEEDED
4939 023232 000766 1$: BR 1$
4940 023234 022626 3$: CMP (SP)+,(SP)+ :CLEAN OFF THE STACK
4941 023236 000207 RTS PC :EXIT
4942
4943 :THIS ROUTINE STORES THE RP11E REGISTERS FOR USE BY THE PROGRAM.
4944 :THE PROGRAM DOES NOT USE THE 'LIVE' REGISTERS FOR ERROR DETERMINATION.
4945 :CALL
4946 :
4947 JSR PC,SAVRP
4948 RETURN
4949
4950
4951
4952
4953
4954 023240 010446 SAVRP: MOV R4,-(SP) :SAVE R4
4955 023242 013704 001324 MOV RPADR,R4 :RP11 ADDRESS
4956 023246 016437 000000 001334 MOV RPOD(R4),$RPOD :STORE RPOD
4957 023254 016437 000002 001336 MOV RPER(R4),$RPER :STORE RPER
4958 023262 016437 000004 001340 MOV RPCS(R4),$RPCS :STORE RPCS

```

```

4950 023270 016437 000006 001342
4951 023270 016437 000010 001344
4952 023270 016437 000012 001346
4953 023270 016437 000014 001350
4954 023270 016437 000016 001352
4955 023270 016437 000024 001354
4956 023270 016437 000026 001356
4957 023270 012604
4958 023270 000207
4959
4960
4961 023346 105737 000041
4962 023352 001005
4963 023354 012702 000144
4964 023360 012701 044166
4965 023366 000404
4966 023366 012702 002734
4967 023372 012701 051546
4968 023372 012703 157776
4969 023376 005737 022232
4970 023402 100402
4971 023410 013703 022476
4972 023414 005046
4973 023416 016616 000002
4974 023422 010166 000002
4975 023426 005737 001322
4976 023432 001006
4977 023434 014341
4978 023436 005302
4979 023440 001375
4980 023442 012737 177777 001322
4981 023442 000207
4982 023450
4983
4984
4985
4986 023452 104412
4987 023454 005737 001322
4988 023460 001430
4989 023462 105737 000041
4990 023466 001005
4991 023470 012702 000144
4992 023474 012701 044166
4993 023500 000404
4994 023502 012702 002734
4995 023506 012701 051546
4996 023512 012703 157776
4997 023516 005737 022232
4998 023522 100402
4999 023524 013703 022476
5000 023530 014143
5001 023532 005302
5002 023534 001375 001322
5003 023536 005037
5004 023542 104413
5005 023544 000207

```

```

MOV RPWC(R4),SRPWC :STORE RPWC
MOV RPBA(R4),SRPBA :STORE RPBA
MOV RPCA(R4),SRPCA :STORE RPCA
MOV RPDA(R4),SRPDA :STORE RPDA
MOV RPM1(R4),SRPM1 :STORE RPM1
MOV SUCA(R4),SSUCA :STORE SUCA
MOV SILO(R4),SSILO :STORE SILO
MOV (SP)+,R4 :RESTORE R4
RTS PC :RETURN

:ROUTINE TO MOVE THE LOADER FROM HIGH MEMORY TO JUST ABOVE THE PROGRAM
MOVLDR: TSTB 41 :WHO LOADED THE PROGRAM
BNE 1$ :BR IF NOT 'ABS'
MOV #100.,R2 :'ABS' LOADER SIZE
MOV #BUFFER+200.,R1 :RELOCATE TO HERE
BR 2$ :FINISH
1$: MOV #1500.,R2 :'XXOP' SIZE
MOV #BUFFER+3000.,R1 :RELOCATE TO HERE (UPPER BOUNDARY)
2$: MOV #157776,R3 :TRY THIS UPPER MEMORY ADDRESS
TST $KT11 :MEMORY MANAGEMENT ?
BMI 3$ :BR IF YES
MOV $LSTAD,R3 :USE ACTUAL HIGH ADDRESS
3$: CLR -(SP) :CLEAR THE STACK
MOV 2(SP),R2 :MOVE THE RETURN ADDRESS DOWN THE STACK
MOV R1,2(SP) :BEGINNING ADDRESS OF BUFFER
TST LDRFLG :LOADER ALREADY RELOCATED ?
BNE 5$ :BR IF YES
4$: MOV -(R3),-(R1) :MOVE A WORD
DEC R2 :COUNT IT
BNE 4$ :BR IF MORE TO GO
5$: MOV #-1,LDRFLG :SET THE LOADER MOVED FLAG
RTS PC :RETURN

:ROUTINE TO RELOCATE THE LOADER TO UPPER MEMORY
RESLDR: SAVREG :SAVE R0 - R5
TST LDRFLG :LOADER MOVED ?
BEQ 5$ :BR IF NOT
TSTB 41 :'ABS' LOADER
BNE 1$ :BR IF NOT
MOV #100.,R2 :'ABS' LOADER SIZE
MOV #BUFFER+200.,R1 :RELOCATE FROM HERE
BR 2$ :FINISH
1$: MOV #1500.,R2 :'XXOP' SIZE
MOV #BUFFER+3000.,R1 :RELOCATE FROM HERE (UPPER BOUNDARY)
3$: MOV #157776,R3 :TRY THIS UPPER MEMORY ADDRESS
TST $KT11 :MEMORY MANAGEMENT ?
BMI 4$ :BR IF YES
MOV $LSTAD,R3 :USE ACTUAL HIGH ADDRESS
4$: MOV -(R1),-(R3) :MOVE A WORD
DEC R2 :COUNT IT
BNE 4$ :BR IF MORE TO GO
5$: CLR LDRFLG :CLEAR THE LOADER MOVED FLAG
RESREG :RESTORE R0 - R5
RTS PC :RETURN

```

```

5006
5007 ;ROUTINE TO WAIT FOR DRIVE READY
5008
5009 02 3546 032764 100000 000000 DRVRDY: BIT #SURDY,RPDS(R4) ;DRIVE READY ?
5010 02 3554 001042 BNE 6$ ;BR IF READY
5011 02 3556 013746 001212 MOV TIMEOUT,-(SP) ;WAIT TIME: APPROX 1 SEC
5012 02 3558 005046 CLR -(SP) ;CLEAR WORKING LOCATION
5013 02 3560 162766 000001 000002 1$: SUB #1,2(SP) ;ANY TIME LEFT IN WAIT LOOP ?
5014 02 3562 001411 BEQ 3$ ;BR IF NOT
5015 02 3564 012716 000144 MOV #100,(SP) ;VALUE FOR APPROX 1 MS
5016 02 3566 032764 100000 000000 2$: BIT #SURDY,RPDS(R4) ;DRIVE READY ?
5017 02 3568 001023 BNE 5$ ;BR IF READY
5018 02 3570 005316 DEC (SP) ;COUNT SOME TIME
5019 02 3572 001372 BNE 2$ ;BR IF NOT FINISHED
5020 02 3574 000763 BR 1$ ;COUNT MAJOR TIME AGAIN
5021 02 3576 004737 023240 JSR PC,SAVRP ;STORE RP11 REGISTERS FOR ERROR MESSAGE
5022 02 3578 012737 023632 001164 3$: MOV #4,$ESCAPE ;ESCAPE TO 4$ ON ERROR
5023 02 3580 104100 ERROR 100 ;DRIVE TIMED OUT
5024 02 3582 005037 001164 4$: CLR $ESCAPE ;CLEAR THE 'ESCAPE' FLAG
5025 02 3584 062706 000004 ADD #4,SP ;CORRECT THE STACK POINTER
5026 02 3586 032777 001000 155270 BIT #SW09,$SWR ;LOOP ON THE ERROR ?
5027 02 3588 001336 BNE DRVRDY ;LOOP IF SET
5028 02 3590 000177 155322 JMP $BYPASS ;GO THE 'BYPASS' ADDRESS
5029 02 3592 062706 000004 5$: ADD #4,SP ;CORRECT THE STACK POINTER
5030 02 3594 000207 6$: RTS PC ;RETURN
5031
5032 ;ROUTINE TO WAIT FOR CONTROLLER READY
5033
5034
5035 02 3664 032764 000200 000004 CONRDY: BIT #RDY,RPCS(R4) ;CONTROLLER READY ?
5036 02 3672 001042 BNE 6$ ;BR IF READY
5037 02 3674 013746 001212 MOV TIMEOUT,-(SP) ;WAIT TIME: APPROX 1 SEC
5038 02 3676 005046 CLR -(SP) ;CLEAR WORKING LOCATION
5039 02 3678 162766 000001 000002 1$: SUB #1,2(SP) ;ANY TIME LEFT IN WAIT LOOP ?
5040 02 3680 001411 BEQ 3$ ;BR IF NOT
5041 02 3682 012716 000144 MOV #100,(SP) ;VALUE FOR APPROX 1 MS
5042 02 3684 032764 000200 000004 2$: BIT #RDY,RPCS(R4) ;CONTROLLER READY ?
5043 02 3686 001023 BNE 5$ ;BR IF READY
5044 02 3688 005316 DEC (SP) ;COUNT SOME TIME
5045 02 3690 001372 BNE 2$ ;BR IF NOT FINISHED
5046 02 3692 000763 BR 1$ ;COUNT MAJOR TIME AGAIN
5047 02 3694 004737 023240 JSR PC,SAVRP ;STORE RP11 REGISTERS FOR ERROR MESSAGE
5048 02 3696 012737 023750 001164 3$: MOV #4,$ESCAPE ;ESCAPE TO 4$ ON ERROR
5049 02 3698 104101 ERROR 101 ;CONTROLLER TIMED OUT
5050 02 3700 005037 001164 4$: CLR $ESCAPE ;CLEAR THE 'ESCAPE' FLAG
5051 02 3702 062706 000004 ADD #4,SP ;CORRECT THE STACK POINTER
5052 02 3704 032777 001000 155152 BIT #SW09,$SWR ;LOOP ON THE ERROR ?
5053 02 3706 001336 BNE CONRDY ;BR IF LOOPING
5054 02 3708 000177 155204 JMP $BYPASS ;'BYPASS' THE TEST
5055 02 3710 062706 000004 5$: ADD #4,SP ;CORRECT THE STACK POINTER
5056 02 3712 000207 6$: RTS PC ;RETURN
5057
5058 ;ROUTINE TO DETERMINE THE NUMBER OF SECTORS ON THE DRIVE BEING TESTED
5059
5060 02 4002 010046 SECNBR: MOV RO,-(SP) ;SAVE RC
5061 02 4004 012737 117230 001244 MOV #4060C,$PAKSIZ ;START WITH THE NUMBER OF SECTORS ON

```

```

5063 024012 005037 001246          CLR      PAKSIZ+2      ;AN RPO2
5064 024016 005000          CLR      RO           ;USE RO AS AN INDEX
5065 024020 113700 002500          MOV     OPB,RO       ;GET THE DRIVE NUMBER
5066 024024 136037 001360 001304  BIT     ATABIT(RO),DRVTYP ;IS THE DRIVE AN RPO3 ?
5067 024028 001404          BEQ     1$           ;BR IF NOT
5068 024032 006337 001244          ASL     PAKSIZ        ;DOUBLE THE SECTOR COUNT
5069 024036 006137 001246          ROL     PAKSIZ+2     ;GET THE CARRY
5070 024040 012600          1$:  MOV     (SP)+,RO   ;RESTORE RO
5071 024044 000207          RTS      PC          ;RETURN

;ROUTINE TO SELECT A PATTERN
5072 024050 010046          PATSEL: MOV     RO, -(SP) ;SAVE RO
5073 024052 005237 001204  1$:  INC     PATNUM       ;INCREMENT THE PATTERN NUMBER
5074 024056 022737 000020 001204  CMP     #16.,PATNUM   ;NUMBER AT MAXIMUM
5075 024060 101002          BHI     2$           ;BR IF NOT
5076 024064 005037 001204          CLR     PATNUM       ;START AT ZERO AGAIN
5077 024068 013700 001204  2$:  MOV     PATNUM,RO    ;SEE IF THE PATTERN IS IN USE
5078 024072 006300          ASL     RO           ;CONVERT INTO A TABLE INDEX
5079 024076 036037 001400 00130C  BIT     BITS(RO),PATRN ;INUSE INDICATOR SET FOR THIS PATTERN ?
5080 024080 001761          BEQ     1$           ;BR IF NOT
5081 024084 012600          MOV     (SP)+,RO    ;RESTORE RO
5082 024088 000207          RTS      PC          ;RETURN

;ROUTINE TO INCREMENT THE DISK ADDRESS
; RETURN + 2 WHEN THE END OF THE PACK IS REACHED
5083 024092 005737 00126C          INCADR: TST     OPRSEL ;USING AN OPERATOR SPECIFIED ADDRESS ?
5084 024096 001066          BNE     5$           ;BR IF YES
5085 024100 005737 001246          TST     PAKSIZ+2     ;SECTORS LEFT ?
5086 024104 003017          BGT     1$           ;BR IF AN UPPER COUNT
5087 024108 005737 001244          TST     PAKSIZ        ;LOWER COUNT AT ZERO ?
5088 024112 001460          BEQ     5$           ;BR IF IT IS
5089 024116 023737 001244 001254  CMP     PAKSIZ,BLKS14 ;ENOUGH SECTORS LEFT FOR A FULL TRANSFER ?
5090 024120 103010          BHS     1$           ;BR IF THERE ARE
5091 024124 013737 001244 002504  MOV     PAKSIZ,$WC   ;USE THE RESIDUE COUNT FOR THIS TRANSFER
5092 024128 000337 002504          SWAB    $WC          ;CONVERT TO A WORD COUNT
5093 024132 005037 001244          CLR     PAKSIZ       ;FORCE COUNT TO ZERO
5094 024136 000410          BR      2$           ;GO AND INCREMENT THE ADDRESS
5095 024140 013737 001256 002504  1$:  MOV     WC14,$WC    ;USE THE STANDARD WORD COUNT
5096 024144 163737 001254 001244  SUB     BLKS14,PAKSIZ ;DECREMENT THE TOTAL SECTORS AVAILABLE
5097 024148 005637 001246          SBC     PAKSIZ+2     ;SUBTRACT ANY CARRY
5098 024152 005046          2$:  CLR     -(SP)       ;CLEAR THE STACK
5099 024156 113716 002512          MOV     $SEC,(SP)   ;GET CURRENT SECTOR
5100 024160 063716 001250          ADD     INCSEC,(SP) ;ADD THE SECTOR INCREMENT
5101 024164 121627 000012          CMPB   (SP),#10.    ;EXCEED THE MAXIMUM ?
5102 024168 002404          BLT     3$           ;BR IF NOT
5103 024172 162716 000012          SUB     #10.,(SP)   ;CORRECT THE RESIDUE
5104 024176 105237 002513          INCB   $TRK         ;INCREMENT THE TRACK
5105 024180 111637 002512          3$:  MOV     (SP),$SEC   ;NEW SECTOR ADDRESS
5106 024184 113716 002513          MOV     $TRK,(SP)  ;GET CURRENT TRACK VALUE
5107 024188 063716 001252          ADD     INCTRK,(SP) ;ADD THE TRACK INCREMENT
5108 024192 121627 000024          CMPB   (SP),#20.    ;EXCEED THE MAXIMUM ?
5109 024196 002404          BLT     4$           ;BR IF NOT
5110 024200 162716 000024          SUB     #20.,(SP)  ;CORRECT THE SIZE
5111 024204 005237 002510          INC     $CYL        ;INCREMENT THE CYLINDER ADDRESS

```


118 024270 112637 002513
 119 024274 000402
 120 024276 062716 000002
 121 024302 000207
 122
 123
 124
 125
 126 0243304 104412
 127 0243306 013701 002506
 128 0243312 005003
 129 0243314 005021
 130 0243316 005021
 131 0243320 010321
 132 0243322 005203
 133 0243324 022703 000012
 134 0243330 001371
 135 0243332 104413
 136 0243334 000207
 137
 138
 139
 140 0243336 104412
 141 0243340 005003
 142 0243342 113703 002512
 143 0243346 001002
 144 0243350 012703 000012
 145 0243354 013702 002504
 146 0243360 013701 002506
 147 0243364 010321
 148 0243366 005302
 149 0243370 001375
 150 0243372 104413
 151 0243374 000207
 152
 153
 154
 155 024376 104412
 156 024400 012701 043657
 157 024404 013702 002504
 158 024410 005003
 159 024412 113711 002513
 160 024416 062701 000002
 161 024422 005302
 162 024424 001372
 163 024426 104413
 164 024430 000207
 165
 166
 167
 168
 169 024432 104412
 170 024434 013700 001204
 171 024440 006300
 172 024442 013701 002506
 173 024446 013702 002504

```

4S:  MOVB  (SP)+,STRK  ;NEW TRACK ADDRESS
      BR    6S          ;EXIT
5S:  ADD   #2,(SP)     ;INCREMENT RETURN - AT END
6S:  RTS   PC          ;RETURN

;ROUTINE TO SETUP A BUFFER WITH THE SEQUENTIAL HEADERS FOR CYLINDER C
; TRACK 0

L0DSEC: SAVREG          ;SAVE R0 - R5
        MOV   $BUF,R1  ;BUFFER ADDRESS
        CLR   R3       ;SECTOR COUNTER
1S:    CLR   (R1)+     ;UPPER WORD OF HEADER
        CLR   (R1)+     ;CYLINDER/TRACK WORD OF HEADER
        MOV   R3,(R1)+ ;SECTOR ADDRESS
        INC   R3       ;INCREMENT THE SECTOR ADDRESS
        CMP   #10,,R3  ;FINISHED ?
        BNE  1S        ;BR IF NOT
        RESREG         ;RESTORE R0 - R5
        RTS   PC       ;RETURN

;FILL THE BUFFER WITH THE SECTOR ADDRESS (CONTAINED IN '$SEC')

FILSEC: SAVREG          ;STORE R0-R5
        CLR   R3       ;CLEAR THE INDEX
        MOVB  $SEC,R3  ;SECTOR ADDRESS
        BNE  1S        ;BR IF NOT SECTOR ZERO
        MOV   #10,,R3  ;USE 10 AS VALUE FOR SECTOR ZERO
1S:    MOV   $WC,R2    ;WORD COUNT
        MOV   $BUF,R1  ;BUFFER START ADDRESS
2S:    MOV   R3,(R1)+  ;MOVE DATA INTO THE BUFFER
        DEC   R2       ;DECREMENT THE COUNTER
        BNE  2S        ;BR IF MORE TO DO
        RESREG         ;RESTORE R0-R5
        RTS   PC       ;RETURN

;ROUTINE TO FILL THE BUFFER WITH THE TRACK ADDRESS (CONTAINED IN 'STRK')

FILTRK: SAVREG          ;STORE R0-R5
        MOV   #BUFFER+1,R1 ;BUFFER ADDRESS
        MOV   $WC,R2    ;WORD COUNT
        CLR   R3       ;CLEAR THE DATA STORAGE
1S:    MOVB  STRK,(R1)  ;MOVE TRACK ADDRESS INTO THE BUFFER
        ADD  #2,R1     ;INCREMENT THE BUFFER POINTER
        DEC   R2       ;DECREMENT THE COUNTER
        BNE  1S        ;BR IF MORE TO DO
        RESREG         ;RESTORE R0-R5
        RTS   PC       ;RETURN

;THIS ROUTINE WILL MOVE THE 16 WORDS OF THE
;DESIRED PATTERN INTO THE DATA BUFFER.

SETBUF: SAVREG          ;SAVE R0 - R5
        MOV   PATNUM,R0 ;GET THE PATTERN NUMBER
        ASL  R0        ;CONVERT NUMBER TO INDEX
        MOV   $BUF,R1  ;FIRST ADDRESS
        MOV   $WC,R2    ;WORD COUNT
  
```

```

1874 024452 016003 001440 1S:  MOV  PAT.PT(R0),R3  ; PICKUP PATTERN POINTER
1875 024456 012704 000020 2S:  MOV  #16,R4          ; PATTERN COUNTER
1876 024460 012321 2S:  MOV  (R3)+,(R1)+    ; MOVE WORD 'N' INTO DATA BUFFER
1877 024464 005302  ; DEC  R2            ; DECREMENT THE COUNTER
1878 024466 001403  ; BEO  R2            ; BR IF DONE
1879 024470 005304  ; DEC  R4            ; DECREMENT THE PATTERN COUNTER
1880 024472 001767  ; BEO  R4            ; START PATTERN AGAIN
1881 024474 000772  ; BR   R2            ; CONTINUE
1882 024476 104413 3S:  RESREG  ; RESTORE R0 - R5
1883 024500 000207  ; RTS  PC           ; RETURN

; THIS ROUTINE LOADS THE PHYSICAL TRANSFER ADDRESS INTO '$BUF'
1884 024502 013737 001224 002506 GETBUF: MOV  ACTMEM,$BUF  ; LOWER 16 BITS OF PHYSICAL ADDRESS
1885 024510 013746 001226  ; MOV  ACTMEM+2,-(SP) ; UPPER ADDRESS BITS ON THE STACK
1886 024514 042716 177774  ; BIC  #103,(SP)      ; LEAVE ONLY THE LOWER 2 BITS
1887 024520 006316  ; ASL  (SP)           ; ALIGN UPPER ADDRESS BITS
1888 024524 006316  ; ASL  (SP)           ; ALIGN UPPER ADDRESS BITS
1889 024528 006316  ; ASL  (SP)           ; ALIGN UPPER ADDRESS BITS
1890 024530 112637 002501  ; MOV  (SP)+,DPB+1    ; LOAD 'MEX' BITS
1891 024534 013746 002504  ; MOV  $WC,-(SP)      ; PUT WORD COUNT ON THE STACK
1892 024540 006316  ; ASL  (SP)           ; DOUBLE THE WORD COUNT
1893 024542 062637 001224  ; ADD  (SP)+,ACTMEM   ; FORM NEXT STARTING ADDRESS
1894 024546 005537 001226  ; ADC  ACTMEM+2       ; ADD ANY CARRY
1895 024550 023737 001226 001236  ; CMP  ACTMEM+2,HIMEM+2 ; EXCEEDING HIGHEST BANK ADDRESS
1896 024560 101005  ; BHI  R5            ; BR IF HIGHER
1897 024562 103413  ; BLO  R5            ; BR IF LOWER
1898 024564 023737 001224 001234  ; CMP  ACTMEM,HIMEM   ; CHECK LOWER 16 BITS
1899 024572 103407  ; BLO  R5            ; BR IF LOWER
1900 024574 013737 001240 001224 1S:  MOV  LOMEM,ACTMEM   ; RESET ADDRESS
1901 024602 013737 001242 001226  ; MOV  LOMEM+2,ACTMEM+2 ; UPPER ADDRESS BITS
1902 024610 000734  ; BR   GETBUF        ; TRY AGAIN
1903 024612 000207 2S:  RTS  PC           ; RETURN

; THIS ROUTINE COMPARES A 16 WORD DATA PATTERN AGAINST THE
; DATA BUFFER. THE PATTERN NUMBER IS IN 'PATNUM'
1904 024614 104412 001222 DATCMP: SAVREG  ; SAVE R0-R5
1905 024616 005737  ; TST  MMACTV        ; MEMORY MANAGEMENT ACTIVE ?
1906 024622 001510  ; BEO  R5            ; BR IF NOT

```

5215	024624	005037	172340		CLR	KIPAR0	: LOAD ADDRESS REGISTER 0
5216	024630	012737	000200	172342	MOV	#200, KIPAR1	: LOAD ADDRESS REGISTER 1
5217	024636	012737	000400	172344	MOV	#400, KIPAR2	: LOAD ADDRESS REGISTER 2
5218	024644	012737	177600	172356	MOV	#177600, KIPAR7	: ACCESS TO I/O PAGE
5219	024652	005046			CLR	-(SP)	: MAKE ROOM ON THE STACK
5220	024654	005046			CLR	-(SP)	: MAKE ROOM ON THE STACK
5221	024656	013766	002506	000002	MOV	\$BUF, 2(SP)	: LOWER 16 BITS OF STARTING ADDRESS
5222	024664	113716	002501		MOV B	DPB+1, (SP)	: UPPER ADDRESS BITS
5223	024670	006216			ASR	(SP)	: RIGHT JUSTIFY THE ADDRESS BITS
5224	024672	006216			ASR	(SP)	: RIGHT JUSTIFY THE ADDRESS BITS
5225	024674	006216			ASR	(SP)	: RIGHT JUSTIFY THE ADDRESS BITS
5226	024676	006216			ASR	(SP)	: RIGHT JUSTIFY THE ADDRESS BITS
5227	024700	012703	000006		MOV	#6, R3	: LOAD COUNTER
5228	024704	006216			ASR	(SP)	: CONVERT ADDRESS TO BLOCK ADDRESS
5229	024706	006066	000002		ROR	2(SP)	: LOWER 16 BITS
5230	024712	005303			DEC	R3	: DECREMENT THE COUNTER
5231	024714	001373			BNE	1\$: CONTINUE UNTIL FINISHED
5232	024716	042766	000177	000002	BIC	#177, 2(SP)	: CLEAR LOWER BLOCK ADDRESS BITS
5233	024724	016637	000002	172346	MOV	2(SP), KIPAR3	: LOAD ADDRESS REGISTER 3
5234	024732	062766	000200	000002	ADD	#200, 2(SP)	: INCREMENT BLOCK ADDRESS BY 4K
5235	024740	016637	000002	172350	MOV	2(SP), KIPAR4	: LOAD ADDRESS REGISTER # 4
5236	024746	012737	077406	172300	MOV	#77406, KIPDR0	: LOAD DESCRIPTOR REGISTER # 0
5237	024754	012737	077406	172302	MOV	#77406, KIPDR1	: LOAD DESCRIPTOR REGISTER # 1
5238	024762	012737	077406	172304	MOV	#77406, KIPDR2	: LOAD DESCRIPTOR REGISTER # 2
5239	024770	012737	077406	172306	MOV	#77406, KIPDR3	: LOAD DESCRIPTOR REGISTER # 3
5240	024776	012737	077406	172310	MOV	#77406, KIPDR4	: LOAD DESCRIPTOR REGISTER # 4
5241	025004	012737	077406	172316	MOV	#77406, KIPDR7	: LOAD DESCRIPTOR REGISTER # 7
5242	025012	023727	002504	010000	CMP	\$WC, #4096	: WAS TRANSFER GREATER THAN 4K ?
5243	025020	101411			BLOS	2\$: BR IF NOT
5244	025022	062766	000200	000002	ADD	#200, 2(SP)	: INCREMENT THE ADDRESS BY 4K
5245	025030	016637	000002	172352	MOV	2(SP), KIPAR5	: LOAD ADDRESS REGISTER # 5
5246	025036	012737	077406	172312	MOV	#77406, KIPDR5	: LOAD DESCRIPTOR REGISTER # 5
5247	025044	013701	002506		MOV	\$BUF, R1	: LOAD STARTING ADDRESS
5248	025050	005737	001222		TST	MMACTV	: MEMORY MANAGEMENT ACTIVE ?
5249	025054	001425			BEQ	CMPAR	: BR IF NOT
5250	025056	042701	140000		BIC	#140000, R1	: CHANGE PHYSICAL BASE TO VIRTUAL BASE
5251	025062	052701	060000		BIS	#60000, R1	: START AT VIRTUAL 12K
5252	025066	062706	000004		ADD	#4, SP	: CORRECT THE STACK POINTER
5253	025072	012737	025110	000004	MOV	#3\$, ERRVEC	: CHANGE THE ERROR VECTOR
5254	025100	012737	000020	172516	MOV	#20, #SR3	: ENABLE 22 BIT ADDRESSING MODE
5255	025106	000402			BR	4\$: BR IF AN 11/70
5256	025110	062706	000004		ADD	#4, SP	: CORRECT THE STACK POINTER
5257	025114	012737	000006	000004	MOV	#ERRVEC+2, ERRVEC	: RESTORE THE ERROR VECTOR
5258	025122	012737	000001	177572	MOV	#1, #SR0	: ENABLE MEMORY MANAGEMENT
5259	025130	013737	001342	025420	CMPAR:	\$RPWC, CMCNT	: WORD COUNT TO WORKING LOCATION
5260	025136	063737	002504	025420	ADD	\$WC, CMCNT	: CALCULATE ACTUAL WORDS TRANSFERED
5261	025144	013737	002510	025422	MOV	\$CYL, CMCYL	: CYLINDER ADDRESS
5262	025152	113737	002513	025426	MOV B	\$TRK, CMTRK	: TRACK ADDRESS
5263	025160	113737	002512	025424	MOV B	\$SEC, CMSEC	: SECTOR ADDRESS
5264	025166	013737	001320	025416	CMSTR:	CMPT, LIMIT	: DISPLAY LIMIT
5265	025174	005237	025416		INC	LIMIT	: CONVERT PARAMETER INTO LIMIT VALUE
5266	025200	010137	025430		MOV	R1, CMBUF	: STARTING ADDRESS OF SECTOR BUFFER
5267	025204	005037	025412		CLR	FRSTER	: CLEAR 'FIRST ERROR' INDICATOR
5268	025210	005037	025414		CLR	ERCTR	: CLEAR ERROR COUNTER
5269	025214	023727	025420	000400	CMP	CMCNT, #256	: IS BUFFER SIZE GREATER THAN ONE SECTOR ?
5270	025222	101003			BHI	1\$: BR IF IT IS

5271	025224	013702	025420		MOV	CMCNT,R2	: LESS THAN, USE REMAINING BUFFER
5272	025230	000402			BR	2\$	
5273	025232	012703	000400	1\$:	MOV	#256,R2	: COMPARE SECTOR
5274	025236	160237	025420	2\$:	SUB	R2,CMCNT	: DECREMENT WORD COUNT
5275	025242	013704	001204		MOV	PATNUM,R4	: PATTERN NUMBER
5276	025246	006304			ASL	R4	: GENERATE INTO AN INDEX
5277	025250	012703	000020	CMDAT:	MOV	#16,R3	: R3 IS PATTERN POSITION COUNTER
5278	025254	016405	001440		MOV	PAT.PT(R4),R5	: PATTERN ADDRESS
5279	025260	022125		1\$:	CMP	(R1)+,(R5)+	: COMPARE BUFFER WITH PATTERN
5280	025262	001406			BEQ	2\$: BR IF EQUAL
5281	025264	032777	000010	153646	BIT	#5W03,2SWR	: SWITCH 3 SET ?
5282	025272	001002			BNE	2\$: BR IF NOT
5283	025274	004737	025432		JSR	PC,CMPRT	: TYPE THE ERROR
5284	025300	005302		2\$:	DEC	R2	: DECREMENT SIZE COUNT
5285	025302	001403			BEQ	3\$: BR WHEN AT END
5286	025304	005303			DEC	R3	: DECREMENT PATT POS COUNT
5287	025306	001364			BNE	1\$: BR IF NOT AT END OF PATT
5288	025310	000757			BR	CMDAT	: RESTART THE PATTERN
5289	025312	004737	025630	3\$:	JSR	PC,ENDCMP	: PRINT LAST LINE (IF ERRORS)
5290	025316	005737	025420		TST	CMCNT	: AT END OF BUFFER
5291	025322	003424			BLE	CMPRX	: BR IF AT END
5292	025324	005237	025424		INC	CMSEC	: INCREMENT SECTOR
5293	025330	023727	025424	000012	CMP	CMSEC,#10.	: SECTOR GREATER THAN MAX ?
5294	025336	103713			BLO	CMSTR	: BR IF NOT GREATER THAN MAX
5295	025340	005037	025424		CLR	CMSEC	: CLEAR SECTOR ADDRESS
5296	025344	005237	025426		INC	CMTRK	: INCREMENT TRACK
5297	025350	023727	025426	000024	CMP	CMTRK,#20.	: TRACK GREATER THAN MAX ?
5298	025356	103703			BLO	CMSTR	: BR IF NOT GREATER
5299	025360	005037	025426		CLR	CMTRK	: RESET TRACK ADDRESS
5300	025364	005237	025422		INC	CMCYL	: INCREMENT CYLINDER ADDRESS
5301	025370	000137	025166		JMP	CMSTR	: CONTINUE WITH COMPARE
5302	025374	104413		CMPRX:	RESREG		: RESTORE R0 - R5
5303	025376	005737	001222		TST	MMACTV	: MEMORY MANAGEMENT ACTIVE ?
5304	025402	001402			BEQ	1\$: BR IF NOT
5305	025404	005037	177572		CLR	2\$SRD	: TURN OFF MEMORY MANAGEMENT
5306	025410	000207		1\$:	RTS	PC	
5307							
5308	025412	000000		FRSTER:	.WORD	0	: FIRST ERROR INDICATOR
5309	025414	000000		ERCTR:	.WORD	0	: NUMBER OF ERRORS
5310	025416	000000		LIMIT:	.WORD	0	: DISPLAY LIMIT
5311	025420	000000		CMCNT:	.WORD	0	: WORD COUNT
5312	025422	000000		CMCYL:	.WORD	0	: CYLINDER ADDRESS
5313	025424	000000		CMSEC:	.WORD	0	: SECTOR ADDRESS
5314	025426	000000		CMTRK:	.WORD	0	: TRACK ADDRESS
5315	025430	000000		CMBUF:	.WORD	0	: BEGINNING ADDRESS OF SECTOR
5316							
5317							
5318							
5319	025432	005737	025412		CMPRT:	TST	FRSTER
5320	025436	001037			BNE	CMPRT1	: BR IF NOT
5321	025440	005737	001222		TST	MMACTV	: MEMORY MANAGEMENT ACTIVE ?
5322	025444	001011		1\$:	BNE	1\$: BR IF ACTIVE
5323	025446	012737	025446	001116	MOV	#.SERRPC	: PC FOR ERROR MESSAGE
5324	025454	112737	000102	001114	MOVB	#102,\$ITEMB	: COMPARISON ERROR MESSAGE NUMBER
5325	025462	004737	017254		JSR	PC,TYPERR	: REPORT THE ERROR
5326	025466	000412			BP	2\$: CONTINUE

```

5327 025470 005037 177572 1S: CLR 2#SRO ;TURN OFF MEMORY MANAGEMENT
5328 025474 012737 025474 001116 MOV #1,SERRPC ;PC FOR ERROR MESSAGE
5329 025502 112737 000103 001114 MOVB #103,$ITEMB ;COMPARISON ERROR MESSAGE WITH MM REGISTERS
5330 025510 004737 017254 JSR PC,TYPERR ;REPORT THE COMPARISON ERROR
5331 025514 012737 177777 025412 2S: MOV #-1,FASTER ;SET FIRST ERROR INDICATOR
5332 025522 005737 001222 TST MMACTV ;MEMORY MANAGEMENT ACTIVE ?
5333 025526 001403 BEQ CMPRT1 ;BR IF 0
5334 025530 012737 000001 177572 MOV #1,2#SRO ;ENABLE MEMORY MANAGEMENT AGAIN
5335 025536 005737 025416 CMPRT1: TST LIMIT ;TIMEOUT LIMIT REACHED ?
5336 025542 001403 BEQ 1S ;BR IF IT HAS
5337 025544 005337 025416 DEC LIMIT ;DECREMENT LIMIT COUNTER
5338 025550 001004 BNE 2S ;BR IF NOT AT LIMIT
5339 025552 032777 000020 153360 1S: BIT #SW04,2SWR ;PRINT ALL DATA COMPARE ERRORS ?
5340 025560 001420 BEQ 3S ;BR IF NOT
5341 025562 010137 001122 2S: MOV R1,$BDAOR ;ADDRESS OF BAD WORD
5342 025566 152737 000002 001122 SUB #2,$BDAOR ;ADJUST ADDRESS
5343 025574 016537 177776 001124 MOV -2(R5),$GDDAT ;GOOD DATA
5344 025602 016137 177776 001126 MOV -2(R1),$BDDAT ;BAD DATA
5345 025610 112737 000104 001114 MOVB #104,$ITEMB ;ERROR TABLE INDEX
5346 025616 004737 017254 JSR PC,TYPERR ;DISPLAY WORD THAT DIDN'T COMPARE
5347 025622 005237 025414 3S: INC ERCTR ;COUNT THE ERROR
5348 025626 000207 RTS PC ;RETURN
5349
5350 ;LAST LINE OF COMPARE ERROR REPORTING
5351
5352 025630 005737 025414 ENDCMP: TST ERCTR ;SEE IF ANY ERRORS
5353 025634 001406 BEQ 2S ;BR IF NONE
5354 025636 012737 025646 001164 MOV #1S,$ESCAPE ;ESCAPE TO 1S ON ERROR
5355 025644 104105 ERROR 105 ;NUMBER OF COMPARE ERRORS
5356 025646 005037 001164 1S: CLR $ESCAPE ;CLEAR THE ERROR ESCAPE ADDRESS
5357 025652 000207 2S: RTS PC ;RETURN
5358
5359 ;ROUTINE TO CLEAR THE RP11E
5360
5361 ;THE FIRST 'CLEAR' TERMINATES ANY ORDER WHICH IS IN PROGRESS. THE SECOND
5362 ;'CLEAR' CLEARS THE 'PROGRAM ERROR' WHICH MAY HAVE BEEN SET BY THE FIRST
5363 ;'CLEAR'.
5364
5365 ;NOTE: R4 MUST CONTAIN THE BASE ADDRESS OF THE RP11E
5366
5367 025654 012764 000001 000004 CLRP: MOV #CLEAR,RPCS(R4) ;CLEAR THE RP11
5368 025662 051616 BIS (SP),(SP) ;DELAY FOR...
5369 025664 051616 BIS (SP),(SP) ;...AT LEAST
5370 025666 051616 BIS (SP),(SP) ;...4 MICROSECONDS.
5371 025670 012764 000001 000004 MOV #CLEAR,RPCS(R4) ;CLEAR THE RP11
5372 025676 000207 RTS PC ;RETURN
5373
5374 ;ROUTINE TO SIZE MEMORY AND TO SETUP FOR TEST 14
5375
5376 025700 012737 000200 022232 SIZMEM: MOV #BIT07,$KT11 ;TELL THE 'SIZE' ROUTINE TO USE MM
5377 025706 004737 022174 6S: JSR PC,$SIZE ;FIND OUT HOW MUCH MEMORY AND IF THE
5378 ;SYSTEM HAS A KT11
5379 025712 005737 022232 TST $KT11 ;DOES THE SYSTEM HAVE A KT11 ?
5380 025716 100060 BPL 2S ;BR IF NOT
5381 025720 023727 022500 001540 CMP $LSTBK,#1540 ;MORE THAN 28K ON THE SYSTEM ?
5382 025726 101003 BHI 1S ;BR IF MORE THAN 28K

```

5383	025730	005037	022232		CLR	\$KT11	: CLEAR MEMORY MANAGEMENT INDICATOR
5384	025734	000764			BR	6\$: DO IT AGAIN WITH MEM. MAN. DESELECTED
5385	025736	012737	157776	001302	1\$: MOV	#157776, MEMSIZ	: ADDRESS OF LAST NON MEMORY MANAGEMENT LOCATION
5386	025744	013737	022500	001230	MOV	\$LSTBK, MAXMEM	: HIGH MEMORY BANK ADDRESS
5387	025752	005037	001232		CLR	MAXMEM+2	: CLEAR UPPER MEMORY ADDRESS BITS
5388	025756	006337	001230		ASL	MAXMEM	: CONVERT THE BANK COUNT INTO AN-
5389	025762	006137	001232		ROL	MAXMEM+2	: ABSOLUTE MEMORY ADDRESS
5390	025766	006337	001230		ASL	MAXMEM	: CONVERT THE BANK COUNT INTO AN
5391	025772	006137	001232		ROL	MAXMEM+2	: ABSOLUTE MEMORY ADDRESS
5392	025776	006337	001230		ASL	MAXMEM	: CONVERT THE BANK COUNT INTO AN
5393	026002	006137	001232		ROL	MAXMEM+2	: ABSOLUTE MEMORY ADDRESS
5394	026006	006337	001230		ASL	MAXMEM	: CONVERT THE BANK COUNT INTO AN
5395	026012	006137	001232		ROL	MAXMEM+2	: ABSOLUTE MEMORY ADDRESS
5396	026016	006337	001230		ASL	MAXMEM	: CONVERT THE BANK COUNT INTO AN
5397	026022	006137	001232		ROL	MAXMEM+2	: ABSOLUTE MEMORY ADDRESS
5398	026026	006337	001230		ASL	MAXMEM	: CONVERT THE BANK COUNT INTO AN
5399	026032	006137	001232		ROL	MAXMEM+2	: ABSOLUTE MEMORY ADDRESS
5400	026036	063737	022476	001230	ADD	\$LSTAD, MAXMEM	: ADD THE SIZE OF THE LAST BANK
5401	026044	005537	001232		ADC	MAXMEM+2	: HANDLE ANY CARRY
5402	026050	012737	020000	001264	MOV	#8192., MAXWC	: SETUP FOR MAXIMUM TRANSFER SIZE
5403	026056	000433			BR	5\$: EXIT
5404	026060	013737	022476	001302	2\$: MOV	\$LSTAD, MEMSIZ	: HIGH NON MEMORY MANAGEMENT ADDRESS
5405	026066	013737	022476	001230	MOV	\$LSTAD, MAXMEM	: HIGH NON MEMORY MANAGEMENT ADDRESS
5406	026074	005037	001232		CLR	MAXMEM+2	: DON'T NEED THE UPPER WORD
5407	026100	013737	001230	001264	MOV	MAXMEM, MAXWC	: CONVERT MEMORY SIZE TO WORD COUNT
5408	026106	162737	043656	001264	SUB	#BUFFER, MAXWC	: SUBTRACT STARTING ADDRESS OF BUFFER
5409	026114	000241			CLC		: CLEAR THE 'C' BIT
5410	026116	006037	001264		ROR	MAXWC	: CHANGE TO WORD COUNT
5411	026122	105737	000041		TSTB	41	: SEE WHICH LOADER
5412	026126	001004			BNE	4\$: BR IF 'XXDP'
5413	026130	162737	000144	001264	3\$: SUB	#100., MAXWC	: 'ABS' LOADER SIZE (PLUS A LITTLE BIT)
5414	026136	000403			BR	5\$: EXIT
5415	026140	162737	002734	001264	4\$: SUB	#1500., MAXWC	: 'XXDP' SIZE
5416	026146	000207			5\$: RTS	PC	: RETURN
5417							
5418							
5419							
5420							
5421							
5422							
5423							
5424	026150	005737	001216		GETADR: TST	BUSADR	: INPUT FROM TTY REQUESTED?
5425	026154	001425			BEQ	3\$: NO-- BRANCH
5426	026156	005037	001216		CLR	BUSADR	: YES-- CLEAR THE REQUEST FLAG
5427	026162	104401	001173		1\$: TYPE	, \$CRLF	: CR-LF
5428	026166	104401	032162		TYPE	, #RPADR	: 'RPADR='
5429	026172	013746	001324		MOV	RPADR, -(SP)	: SAVE RPADR FOR TYPEOUT
5430							: RP11 ADDRESS
5431	026176	104403			TYPOS		: GO TYPE--OCTAL ASCII
5432	026200	006			.BYTE	6	: TYPE 6 DIGIT(S)
5433	026201	001			.BYTE	1	: TYPE LEADING ZEROS
5434	026202	104401	031322		TYPE	, SLASH	: '/'
5435	026206	104411			RDLIN		: GET THE ENTRY
5436	026210	012601			MOV	(SP)+, R1	: STARTING ADDRESS OF INPUT ASCII STRING
5437	026212	004037	027136		JSR	RO, CK.NUM	: CHECK THE NUMBER
5438	026216	026162			1\$:		: ILLEGAL INPUT

: THIS ROUTINE IS USED TO ENSURE THAT THE BUS ADDRESS
 : OF THE RP11 IS SETUP TO READ THE PROPER VALUE.
 : IT WILL ALSO READ THE ADDRESS FROM THE TTY IF
 : REQUIRED.
 : NOTE: THIS ROUTINE DESTROYS R0 - R4

TRAP TABLE

5439	026220	026224			2\$: TERMINATED WITH A 'CR'
5440	026222	026230			3\$: NO ENTRY - 'CR' ONLY
5441	026224	010237	001324		2\$:	MOV	R2 RPADR		: STORE THE ADDRESS
5442	026230	013701	000004		3\$:	MOV	ERRVEC,R1		: SAVE THE ERROR VECTOR
5443	026234	012737	026254	000004		MOV	#4\$,ERRVEC		: SETUP FOR TRAP
5444	026242	005777	153056			TST	RPADR		: CHECK FOR RP11
5445	026244	010137	000004			MOV	R1 ,ERRVEC		: RESTORE ERROR VECTOR
5446	026246	000207				RTS	PC		: RETURN
5447	026254	010137	000004		4\$:	MOV	R1 ERRVEC		: RESTORE ERROR VECTOR
5448	026260	026266				CMP	(SP)+,(SP)+		: CLEAN OFF THE STACK
5449	026262	012737	026272	001164		MOV	#5\$,SE\$CAPE		: ESCAPE TO 5\$ ON ERROR
5450	026270	104001				ERROR	1		: REPORT THE ERROR
5451	026272	005037	001164		5\$:	CLR	SE\$CAPE		: CLEAR ERROR ESCAPE ADDRESS
5452	026276	005737	000042			TST	#42		: IS THERE A MONITOR?
5453	026302	001727				BEQ	1\$: NO--GO ASK FOR ADDRESS
5454	026304	005037	016776			CLR	SEOPCT		: NO PASSES
5455	026310	000137	016616			JMP	SEOP		: GO TO END OF PROGRAM

: THIS ROUTINE ALLOWS THE OPERATOR TO CONTROL THE PROGRAM
(CONVERSATION MODE)

5456									
5457									
5458									
5459									
5460	026314	104401	031324		ENTPRM:	TYPE	,ENTNM		: 'ENTER TEST NUMBER'
5461	026320	104411				RDLIN			: GET THE ENTRY
5462	026322	012601				MOV	(SP)+,R1		: STARTING ADDRESS OF INPUT ASCII STRING
5463	026324	004037	027136			JSR	RD,CK,NUM		: CHECK THE NUMBER
5464	026330	026314				ENTPRM			: ILLEGAL INPUT
5465	026332	026336				1\$: TERMINATED WITH A 'CR'
5466	026334	026362				2\$: NO ENTRY - 'CR' ONLY
5467	026336	022702	000020		1\$:	CMP	#STN,R2		: VALID NUMBER ?
5468	026342	101764				BLOS	ENTPRM		: BR IF NOT
5469	026344	006302				ASL	R2		: CONVERT INTO A BIT INDEX
5470	026346	016237	001400	001276		MOV	BITS(R2),TSTNMS		: TEST SELECTION BIT
5471	026354	006202				ASR	R2		: CHANGE BACK INTO A WORD COUNT
5472	026356	010246				MOV	R2 -(SP)		: SAVE THE TEST NUMBER ENTRY
5473	026360	000405				BR	ENTPRI		: GET THE DRIVE NUMBER
5474	026362	012737	017777	001276	2\$:	MOV	#17777,TSTNMS		: SELECT ALL TESTS
5475	026370	012746	177777			MOV	#-1,-(SP)		: ALL TESTS SELECTED
5476	026374	005037	001312		ENTPRI:	CLR	DRVSEL		: CLEAR DRIVE SELECTION WORD
5477	026400	004737	022566			JSR	PC,RPINIT		: CHECK THE DRIVE STATUS
5478	026404	012737	000312	001210		MOV	#202,MAXCYL		: ASSUME ONLY RPO2'S
5479	026412	005737	001304			TST	DRVTP		: SEE WHICH TYPE
5480	026416	001403				BEQ	1\$: BR IF ONLY RPO2'S
5481	026420	012737	000625	001210		MOV	#405,MAXCYL		: RPO3'S
5482	026426	104401	031352		1\$:	TYPE	,DRVNM		: ENTER DRIVE NUMBER
5483	026432	104411				RDLIN			: GET THE ENTRY
5484	026434	012601				MOV	(SP)+,R1		: STARTING ADDRESS OF INPUT ASCII STRING
5485	026436	004037	027136			JSR	RD,CK,NUM		: CHECK THE NUMBER
5486	026442	026426				1\$: ILLEGAL INPUT
5487	026444	026450				2\$: TERMINATED WITH A 'CR'
5488	026446	026524				5\$: NO ENTRY - 'CR' ONLY
5489	026450	022702	000010		2\$:	CMP	#8, R2		: VALID ENTRY ?
5490	026454	101764				BLOS	1\$: BR IF NOT
5491	026456	116237	001360	001312		MOV	ATABIT(R2),DRVSEL		: DRIVE SELECTION BIT
5492	026464	105762	001370			TST	DRVSTA(R2)		: IS DRIVE ONLINE
5493	026470	003035				BGT	8\$: BR IF ONLINE
5494	026472	104401	031377			TYPE	,DRIVE		: 'DRIVE'

001370		TSTB	DRVSTA(R2)	CHECK DRIVE STATUS AGAIN
027312		BMI	38	BR IF UNSAFE
027276		TYPE	OFFLIN	OFFLINE
001173	38:	BR	48	CONTINUE
	48:	TYPE	UNSAFE	UNSAFE
		TYPE	SCRLF	CR-LF
	58:	BR	ENTPR1	TRY AGAIN
		CLR	R1	CLEAR R1 - USE IT AS AN INDEX
000015		TST	(SP)	ALL TESTS SELECTED ?
		BMI	65	BR IF THEY WERE
		CMQ	815,(SP)	TEST 15, 16, OR 17 SELECTED ?
001370	68:	BLOS	16	BR IF YES SPECIFIC DRIVE NUMBER REQUIRED
		TSTB	DRVSTA(R1)	IS THE DRIVE ONLINE ?
001360	78:	BFB	78	BR IF NOT
		TSTB	ATABIT(R1),DRVSEL	SET THE SELECT BIT FOR THE DRIVE
000010		INC	R1	INCREMENT THE DRIVE INDEX
		CMQ	R1,88	REACHED MAXIMUM ?
000014	88:	BLO	68	BR IF NOT
		CMQ	814,(SP)+	IS TEST 14 SELECTED ?
031406	ENTPR2:	ENTPRX		BR IF NOT
		TYPE	CHNGPM	CHANGE PARAMETERS FOR TEST 14 ?
000131		ROLIN		GET THE RESPONSE
		CMQB	2(SP)+,8'Y	'Y' ENTERED ?
031447		BNE	ENTPRX	BR IF NOT
001264		TYPE	MAXLWD	MAXIMUM WORD COUNT FOR TEST 14:
		MOV	MAXWC,-(SP)	SAVE MAXWC FOR TYPEOUT
				MAXIMUM WORD COUNT FOR TEST 14
				GO TYPE--OCTAL ASCII
				TYPE 5 DIGIT(S)
				SUPPRESS LEADING ZEROS
				CR-LF
001173	18:	TYPE	SCRLF	CR-LF
031523		TYPE	ENTHC	ENTER WORD COUNT FOR TEST 14
		ROLIN		GET THE ENTRY
027136		MOV	(SP)+,R1	STARTING ADDRESS OF INPUT ASCII STRING
		JSR	RD,CK.NUM	CHECK THE NUMBER
				ILLEGAL INPUT
				TERMINATED WITH A 'CR'
				NO ENTRY - 'CR' ONLY
001264	28:	CMQ	R2,MAXWC	VALUE TOO LARGE ?
		BMI	18	BR IF TOO LARGE
001262		CMQ	R2,LMTBK	NEW WORD COUNT LARGER THAN BK ?
		BMI	18	BR IF LARGER
001256		MOV	R2,WC14	LOAD NEW WORD COUNT
031565	ENTPR3:	TYPE	CHNGPT	ENTER PATTERN CODE
		ROLIN		READ THE ENTRY
027136		MOV	(SP)+,R1	STARTING ADDRESS OF INPUT ASCII STRING
		JSR	RD,CK.NUM	CHECK THE NUMBER
		ENTPR3		ILLEGAL INPUT
		IS		TERMINATED WITH A 'CR'
				NO ENTRY - 'CR' ONLY
001300	18:	MOV	R2,PATRN	PATTERN CODE
031640	ENTPR4:	TYPE	CHNGAD	USE A SPECIFIC DISK ADDRESS ?
		ROLIN		READ THE ENTRY
000131		CMQB	2(SP)+,8'Y	'Y' ENTERED ?
		BNE	ENTPRX	BR IF NOT
031713		TYPE	ONEPAT	A SINGLE DATA PATTERN MUST HAVE BEEN SPECIFIED:

5581	027072	104401	032054	1S:	TYPE	.ENTCYL	: IF NOT, PATTERN 0 (ALL ZEROS) WILL BE JSEC'
5582	027073	104411			RDL IN		: ENTER CYLINDER ADDRESS'
5583	027074	012601			MOV	(SP)+,R1	: GET THE ENTRY
5584	027075	004037	027136		JSR	RO,CK.NUM	: STARTING ADDRESS OF INPUT ASCII STRING
5585	027076	026772					: CHECK THE NUMBER
5586	027077	027014					: ILLEGAL INPUT
5587	027078	027050					: TERMINATED WITH A 'CR'
5588	027079	020227	001210	2S:	CMP	R2,MAXCYL	: NO ENTRY - 'CR' ONLY
5589	027080	101364			BHI	1S	: TOO LARGE ?
5590	027081	010237	001266		MOV	R2,SCYL	: BR IF TOO LARGE
5591	027082	104401	032105	3S:	TYPE	.ENTRCK	: LOAD CYLINDER ADDRESS
5592	027083	104411			RDL IN		: ENTER TRACK ADDRESS'
5593	027084	012601			MOV	(SP)+,R1	: GET THE ENTRY
5594	027085	004037	027136		JSR	RO,CK.NUM	: STARTING ADDRESS OF INPUT ASCII STRING
5595	027086	026772					: CHECK THE NUMBER
5596	027087	027014					: ILLEGAL INPUT
5597	027088	027050					: TERMINATED WITH A 'CR'
5598	027089	020227	000023	4S:	CMP	R2,#19.	: NO ENTRY - 'CR' ONLY
5599	027090	101364			BHI	3S	: VALID ENTRY ?
5600	027091	010237	001270		MOV	R2,STRK	: BR IF NOT
5601	027092	104401	032133	5S:	TYPE	.ENTSEC	: TRACK ADDRESS
5602	027093	104411			RDL IN		: 'ENTER SECTOR ADDRESS'
5603	027094	012601			MOV	(SP)+,R1	: STARTING ADDRESS OF INPUT ASCII STRING
5604	027095	004037	027136		JSR	RO,CK.NUM	: CHECK THE NUMBER
5605	027096	026772					: ILLEGAL INPUT
5606	027097	027050					: TERMINATED WITH A 'CR'
5607	027098	020227	000011	6S:	CMP	R2,#9.	: NO ENTRY - 'CR' ONLY
5608	027099	101364			BHI	5S	: VALID ENTRY ?
5609	027100	010237	001272		MOV	R2,SSEC	: BR IF NOT
5610	027101	0.2737	177777		MOV	#-1,OPRSEL	: LOAD SECTOR ADDRESS
5611	027102	000207	001260		MOV	PC	: SET ADDRESS FLAG
5612				ENTPRX:	RTS		: RETURN

: THIS ROUTINE IS USED TO CHECK IF AN ASCII CHARACTER IS A DIGIT
: BETWEEN 0 AND 7.

5581					MOV	#ADR,R1	: ADDRESS OF ASCII CHARACTER
5582					JSR	RO,CK.OCT	: CHECK THE CHARACTER
5583					RETURN1		: CHARACTER IS IN R2 AS A
5584							: OCTAL DIGIT
5585					RETURN2		: CHARACTER IS NOT BETWEEN 0-7

5591	027072	121127	000060	CK.OCT:	CMPB	(R1),#0	: LESS THAN ZERO?
5592	027076	103407			BLO	1S	: YES -- BRANCH
5593	027100	121127	000067		CMPB	(R1),#7	: GREATER THAN SEVEN?
5594	027104	101004			BHI	1S	: YES -- BRANCH
5595	027106	111102			MOVB	(R1),R2	: GET THE CHARACTER
5596	027110	042702	177770		BIC	#7,R2	: STRIP AWAY THE ASCII
5600	027114	000401			BR	2S	: BYPASS RETURN ADJUST
5601	027116	005720		1S:	TST	(RO)+	: ADJUST FOR RETURN
5602	027120	000200		2S:	RTS	RO	: RETURN

: THIS ROUTINE WILL CHECK AN ASCII CHARACTER TO
: DETERMINE WHAT IT IS.

5581
5582
5583
5584
5585
5586
5587
5588
5589
5590
5591
5592
5593
5594
5595
5596
5597
5598
5599
5600
5601
5602
5603
5604
5605
5606

TRAP TABLE

5607
5608
5609
5610
5611
5612 027122 105711
5613 027122 001002
5614 027122 005720
5615 027122 005201
5616 027122 011000
5617 027122 000200
5618
5619
5620
5621
5622
5623
5624
5625
5626
5627
5628
5629 027136 010346
5630 027140 005003
5631 027142 004037 027072
5632 027146 000404
5633 027150 004037 027122
5634 027154 027230
5635 027156 027222
5636 027160 005201
5637 027162 006303
5638 027164 103421
5639 027166 006303
5640 027170 103417
5641 027172 006303
5642 027174 103415
5643 027176 060203
5644 027200 004037 027072
5645 027204 000765
5646 027206 010302 25:
5647 027210 005003 35:
5648 027212
5649 027212 004037 027122
5650 027216 027230
5651 027220 027224
5652 027222 005723 85:
5653 027224 005723 55:
5654 027226 060300
5655 027230 012603 65:
5656 027232 011000
5657 027234 000200
5658
5659
5660
5661 027236 005015 042012 044522
5662 027244 042526 051440 040524

```

:      MOV      #ADR,R1      ; ADDRESS OF ASCII CHARACTER
:      JSR      RD,CK.CHR     ; CHECK CHARACTER
:      RETURN   ADR1         ; UNKNOWN CHARACTER
:      RETURN   ADR2         ; CARRIAGE RETURN * (R1)=ADR+1

CK.CHR: TSTB    (R1)         ; "CARRIAGE RETURN"?
:      BNE     IS           ; NO -- BRANCH
:      TST    (RD)+         ; CARRIAGE RETURN
:      INC    R1            ; MOVE POINTER TO NEXT CHARACTER
IS:     MOV     (RD),R0      ; UNKNOWN CHARACTER
:      RTS     R0           ; RETURN

```

: THIS ROUTINE CHECKS AN ASCII STRING FOR LEGAL CHARACTERS
: AND FORMS AN OCTAL NUMBER IN R2

```

:      MOV      #ADR,R1      ; ADDRESS OF ASCII STRING
:      JSR      RD,CK.NUM     ; GO FORM THE NUMBER
:      RETURN   ADR1         ; ILLEGAL CHARACTER IN THE INPLT STRING
:      RETURN   ADR2         ; "CR" -- R2=NUMBER
:      RETURN   ADR3         ; "CR" WAS FIRST ENTRY

CK.NUM: MOV     R3, -(SP)    ; SAVE R3
:      CLR     R3           ; START NUMBER AT ZERO
:      JSR     RD,CK.OCT    ; OCTAL DIGIT?
:      BNE     IS           ; YES -- BRANCH
:      JSR     RD,CK.CHR    ; CHECK ONE CHARACTER
:      BNE     IS           ; ILLEGAL CHARACTER
:      BNE     IS           ; CARRIAGE RETURN
IS:     INC     R1          ; MOVE TO NEXT CHARACTER
:      ASL    R3           ; FOR THE OCTAL NUMBER IN R3
:      BCS    R3           ; DON'T LET IT GET TO BIG

:      MOV     R3,R3       ; IS THIS AN OCTAL DIGIT?
:      BR     IS           ; YES -- MAKE IT PART OF THE NUMBER
25:    MOV     R3,R2       ; SAVE THE OCTAL NUMBER
:      CLR     R3         ; START WITH ZERO INDEX

35:    JSR     RD,CK.CHR   ; CHECK ONE CHARACTER
:      BNE     IS         ; ILLEGAL CHARACTER
:      BNE     IS         ; CARRIAGE RETURN
:      TST    (R3)+       ; "CR" ONLY
85:    TST    (R3)+
55:    ADD    R3,R0       ; YES -- SAVE THE OCTAL NUMBER
65:    MOV    (SP)+,R3    ; RESTORE R3
:      MOV    (RD),R0     ; PICKUP EXIT ADDRESS
:      RTS    R0         ; RETURN

```

.SBT*L TELETYPE MESSAGES

DRSTAT: .ASCII (15)(12)12)DRIVE STATUS:2(15)(12)(12)

TELETYPE MESSAGES

5663	027252	052524	035123	005015	
5664	027252	000012			
5665	027252	020040	020040	047440	ONLINE: .ASCIZ @ ONLINE@
5666	027270	047111	047111	000105	
5667	027270	020040	020040	052440	UNSAFE: .ASCIZ @ UNSAFE@
5668	027307	051516	043101	000105	
5669	027312	020040	020040	047440	OFFLIN: .ASCIZ @ OFFLINE@
5670	027320	043101	04514	042516	
5671	027327	000			
5672	027337	0400	051040	030120	RP02: .ASCIZ @ RP02@
5673	027337	000062			
5674	027336	020040	050122	031460	RP03: .ASCIZ @ RP03@
5675	027344	000			
5676	027345	041094	044522	042526	DRVTST: .ASCIZ @DRIVE(S) TO BE USED @
5677	027352	051450	020051	047524	
5678	027360	041040	020105	051525	
5679	027366	042105	000040		
5680	027372	047516	042516	000	NONE: .ASCIZ @NONE@
5681	027377	054	000		COMMA: .ASCIZ @,@
5682	027401	015	005012	042524	TSTING: .ASCIZ <15><12><12>@TESTING WITH DRIVE @
5683	027406	052123	047111	020107	
5684	027414	044527	044124	042040	
5685	027422	044522	042526	000040	
5686	027430	020040	000		BLNKS2: .ASCIZ @ @
5687					
5688	027433	015	005012	047520	MSG15A: .ASCIZ <15><12><12>@POWER FAIL TEST@
5689	027440	042527	020122	040506	
5690	027446	046111	052040	051505	
5691	027454	000124			
5692	027456	005015	052012	051125	MSG15B: .ASCII <15><12><12>@TURN CPU POWER O F F AFTER THIS MESSAGE. WAIT@<15><12>
5693	027464	020116	050103	020125	
5694	027472	047527	042527	020122	
5695	027500	047440	043040	043040	
5696	027506	020040	043101	042524	
5697	027514	020122	044124	051511	
5698	027522	046440	051505	040523	
5699	027530	042507	020054	040527	
5700	027536	052111	085015		
5701	027542	020065	042523	047503	.ASCIZ @5 SECONDS, THEN TURN CPU POWER O N AGAIN@<15><12>
5702	027550	042116	026123	052040	
5703	027556	042510	020116	052524	
5704	027564	047122	041440	052520	
5705	027572	050040	053517	051105	
5706	027600	020040	020117	020116	
5707	027606	040440	040507	047111	
5708	027614	005015	000		
5709	027617	015	042412	042116	MSG15C: .ASCIZ <15><12>@END OF POWER FAIL TEST@<15><12>
5710	027624	047440	020106	047520	
5711	027632	042527	020122	040506	
5712	027640	046111	052040	051505	
5713	027646	006524	000012		
5714	027652	005015	050012	047522	MSG15D: .ASCIZ <15><12><12>@PROGRAM WILL LOOP, WAITING FOR DRIVE TO COME ONLINE@<15><12>
5715	027660	051107	046501	053440	
5716	027666	046111	020114	047514	
5717	027674	050117	020054	040527	
5718	027702	052111	047111	020107	

```

771 0277710 027506 020123 051104
772 0277716 053111 020105 047524
773 0277724 021440 046517 020105
774 0277732 027117 04514 042516
775 0277740 005015 000
776 0277743 015 005012 042510
777 0277750 022101 005012 044514
778 0277756 027107 005012 052116
779 0277764 025140 005012 044524
780 0277772 005015 005012 047117
781 030002 025140 005012 040440
782 0300010 025140 005012 042520
783 0300016 025140 005012 044524
784 0300032 025140 005012 031461
785 0300040 025140 005012 040440
786 0300046 025140 005012 040440
787 0300054 025140 005012 040440
788 0300062 025140 005012 040440
789 0300070 025140 005012 046120
790 0300076 027101 005012 051050
791 030101 015 020010 052040
792 030106 023105 005012 020105
793 030114 020117 005012 041511
794 030122 020101 005012 047105
795 030130 025140 005012 046440
796 030138 025140 005012 046440
797 030146 027101 005012 020114
798 030152 027101 005012 044124
799 030160 027506 005012 053111
800 030166 020105 005012 052116
801 030202 027120 005012 047524
802 030212 015 005012 051440
803 030220 024527 005012 020110
804 030226 0220067 005012 020122
805 030234 0220067 005012 051440
806 030242 026105 005012 044524
807 030250 027117 005012 000
808 030255 015 005012 052116
809 030262 021105 024404 040505
810 030270 020104 042101 051104
811 030276 051505 020123 044450
812 030304 020116 041517 040524
813 030312 024514 020072 000
814 030317 015 005012 050122
815 030324 030461 020105 047503
816 030332 052116 047522 020114
817 030340 040520 042516 020114
818 030346 053523 052111 044103
819 030354 052040 051505 000124
820 030362 005015 054524 042520

```

MSG16A: .ASCII (15)(12)(12)HEAD ALIGNMENT ROUTINE

.ASCII (15)(12)(12)CONNECT A JUMPER BETWEEN PINS J13M2 & J13C2 ON THE RP11E BA

.ASCII (15)(12) (REFER TO THE APPLICABLE MAINTENANCE MANUAL FOR THE DRIVE SET)

.ASCIZ (15)(12)(12)TOGGLE SWITCH 7 FOR HEAD SELECTIONS(15)(12)

MSG16B: .ASCIZ (15)(12)ENTER HEAD ADDRESS (IN OCTAL):

MSG17A: .ASCIZ (15)(12)(12)RP11E CONTROL PANEL SWITCH TEST

MSG17B: .ASCIZ (15)(12)TYPE A 'CR' TO START TEST AFTER EACH SETUP(15)(12)

```

5766 0330370 051103
5767 0330370 051103
5768 0330370 051103
5769 0330370 051103
5770 0330370 051103
5771 0330370 051103
5772 0330370 051103
5773 0330370 051103
5774 0330370 051103
5775 0330370 051103
5776 0330370 051103
5777 0330370 051103
5778 0330370 051103
5779 0330370 051103
5780 0330370 051103
5781 0330370 051103
5782 0330370 051103
5783 0330370 051103
5784 0330370 051103
5785 0330370 051103
5786 0330370 051103
5787 0330370 051103
5788 0330370 051103
5789 0330370 051103
5790 0330370 051103
5791 0330370 051103
5792 0330370 051103
5793 0330370 051103
5794 0330370 051103
5795 0330370 051103
5796 0330370 051103
5797 0330370 051103
5798 0330370 051103
5799 0330370 051103
5800 0330370 051103
5801 0330370 051103
5802 0330370 051103
5803 0330370 051103
5804 0330370 051103
5805 0330370 051103
5806 0330370 051103
5807 0330370 051103
5808 0330370 051103
5809 0330370 051103
5810 0330370 051103
5811 0330370 051103
5812 0330370 051103
5813 0330370 051103
5814 0330370 051103
5815 0330370 051103
5816 0330370 051103
5817 0330370 051103
5818 0330370 051103
5819 0330370 051103
5820 0330370 051103
5821 0330370 051103
5822 0330370 051103
5823 0330370 051103
5824 0330370 051103
5825 0330370 051103
5826 0330370 051103
5827 0330370 051103
5828 0330370 051103
5829 0330370 051103
5830 0330370 051103

```

MSG17C: .ASCIZ (15)(12)DISABLE DRIVE # 2

MSG17D: .ASCIZ (15)(12)ENABLE DRIVE # 2

MSG17E: .ASCIZ (15)(12)ASET 'READ ONLY' ON DRIVE 2

MSG17F: .ASCIZ (15)(12)ASET 'READ-WRITE' ON DRIVE 2

MSG17G: .ASCII (15)(12)ASET 'WRITE LOCKOUT' AND CLEAR THE LCA SWITCHES

.ASCIZ (15)(12)ON THE RP11E CONTROL PANELS

MSG17H: .ASCIZ (15)(12)ASET THE FOLLOWING OCTAL VALUES IN THE CYLINDER LCA SWITCHES: 2

MSG17I: .ASCIZ (15)(12)ASET SWITCHES TO 2

MSG17J: .ASCII (15)(12)ARESET THE CYLINDER LCA SWITCHES

5831	031060	012504	020123	047514
5832	031066	020101	053523	052111
5833	031074	004103	051505	
5834	031108	005015	051412	052105
5835	031110	005204	051510	043040
5836	031111	005617	051514	044527
5837	031112	005316	051440	052103
5838	031113	005610	053040	046101
5839	031113	005610	050123	047111
5840	031114	005204	052510	042040
5841	031115	005204	052510	046040
5842	031116	005127	051452	045227
5843	031116	005127	051452	035123
5844	031117	000040		
5845	031117	000040	047412	042116
5846	031120	007440	020106	047503
5847	031121	005116	046117	050040
5848	031122	007101	046105	051440
5849	031122	004527	041524	020110
5850	031123	005204	052123	
5851	031124	005015	052522	052524
5852	031124	007122	051040	030520
5853	031125	007411	041440	047117
5854	031126	005112	046117	050040
5855	031127	007101	046105	051440
5856	031127	004527	041524	042510
5857	031130	020123	047524	047040
5858	031131	005117	040515	006514
5859	031132	000012		
5860				
5861	031132	000057		SLASH: .ASCIZ 0/0
5862	031132	005015	047105	042524
5863	031132	020122	042524	052123
5864	031134	047040	046525	042502
5865	031134	035122	000040	
5866	031135	047105	042524	020122
5867	031136	005110	053111	020105
5868	031136	052516	041115	051105
5869	031137	020072	0000	
5870	031137	00104	044522	042526
5871	031140	000040		
5872	031140	044103	047101	042507
5873	031141	050040	051101	046501
5874	031142	052105	051105	020123
5875	031143	047506	020122	042524
5876	031143	052123	030440	020064
5877	031144	020077	0000	
5878	031144	00115	054101	046511
5879	031145	046525	053440	051117
5880	031146	020104	047503	047125
5881	031147	020124	047506	020122
5882	031147	042524	052123	030440
5883	031150	020064	044450	020116
5884	031151	041517	040524	024514
5885	031152	020072	0000	
5886	031152	00105	052116	051105

.ASCIZ (15)(12) 12)SET THE FOLLOWING OCTAL VALUES IN THE DRIVE LCA SWITCHES: 2

MSG174: .ASCII (15)(12)(12)END OF CONTROL PANEL SWITCH TEST3

.ASCIZ (15)(12)RETURN RP11E CONTROL PANEL SWITCHES TO NORMAL3 15 12.

ENTNM: .ASCIZ (15)(12)ENTER TEST NUMBER: 2

DRVNM: .ASCIZ ENTER DRIVE NUMBER: 2

DRIVE: .ASCIZ 2DRIVE 2

CHNGPM: .ASCIZ 2CHANGE PARAMETERS FOR TEST 14 ? 2

MXWRDS: .ASCIZ 2MAXIMUM WORD COUNT FOR TEST 14 (IN OCTAL): 2

ENTWC: .ASCIZ 2ENTER NEW WORD COUNT (IN OCTAL): 2

```

9887 031530 047040 053505 053440
9888 031536 051117 020104 047503
9889 031544 047125 020124 044450
9890 031552 020116 041517 040524
9891 031560 045114 020072 000000
9892 031566 051116 042101 051105
9893 031572 051101 042101 042105
9894 031578 051440 044524 046105
9895 031600 041505 044524 047117
9896 031614 041440 042117 020105
9897 031622 041750 020122 042524
9898 031630 052123 030440 035064
9899 031636 000040
9900 031640 051523 020105 020101
9901 031646 050123 041505 043111
9902 031654 041511 042040 051511
9903 031662 020113 042101 051104
9904 031670 051505 020123 047506
9905 031676 020122 042524 052123
9906 031704 030440 020064 020077
9907 031712 000000
9908 031713 0040 040450 051440
9909 031720 047111 046107 020105
9910 031726 040504 040524 050040
9911 031734 052101 042524 047122
9912 031742 046440 051525 020124
9913 031750 040510 042526 041040
9914 031756 042505 020116 050123
9915 031764 041505 043111 042511
9916 031772 06504 00012
9917 031775 040 043111 047040
9918 032002 052117 020054 040520
9919 032010 052124 051105 020116
9920 032016 020060 040450 046114
9921 032024 055040 051105 051517
9922 032032 020051 044527 046114
9923 032040 041040 020105 051525
9924 032046 042105 006451 000012
9925 032054 047105 042524 020122
9926 032062 054503 044514 042116
9927 032070 051105 040440 042104
9928 032076 042522 051523 020072
9929 032104 000000
9930 032105 052105 052116 051105
9931 032112 052040 040522 045503
9932 032120 040440 042104 042522
9933 032126 051523 020072 000000
9934 032133 000000 052116 051105
9935 032140 051440 041505 047524
9936 032146 020122 042101 051104
9937 032154 051505 035123 000040
9938
9939 032162 050122 042101 020122
9940 032170 020075 000000

```

CHNGPT: .ASCIZ 2ENTER PATTERN SELECTION CODE FOR TEST 14: 2

CHNGAD: .ASCIZ 2USE A SPECIFIC DISK ADDRESS FOR TEST 14: 2

ONEPAT: .ASCII 2 (A SINGLE DATA PATTERN MUST HAVE BEEN SPECIFIED) 15: 12

.ASCIZ 2 IF NOT, PATTERN 0 (ALL ZEROS) WILL BE USED 15: 12

ENTCYL: .ASCIZ 2ENTER CYLINDER ADDRESS: 2

ENTTRK: .ASCIZ 2ENTER TRACK ADDRESS: 2

ENTSEC: .ASCIZ 2ENTER SECTOR ADDRESS: 2

MRPADR: .ASCIZ 2RPADR = 2

.SBTL EPROC MESSAGES

ERROR MESSAGES

5999	032663	051116	047117	055055	EM12: .ASCII	NON-ZERO DATA IN PART OF SECTOR WHICH SHOULD HAVE<'15','12'
6000	032670	051105	020117	040504		
6001	032676	040524	044440	020116		
6002	032704	040520	052122	047440		
6003	032712	020106	044523	052103		
6004	032720	051117	053440	044510		
6005	032726	044103	051440	047510		
6006	032734	046123	020104	040510		
6007	032742	042526	050105			
6008	032746	042502	047105	055040	.ASCIZ	BEEN ZERO FILLED DURING PARTIAL WRITE
6009	032754	051105	020117	044506		
6010	032762	046114	042105	042040		
6011	032770	051125	047111	020107		
6012	032776	040520	052122	040511		
6013	033004	020114	051127	052111		
6014	033012	000105				
6015	033014	042447	050117	020047	EM13: .ASCII	'EOP' DIDN'T SET DURING A 2 SECTOR WRITE BEGINNING
6016	033022	044504	047104	052047		
6017	033030	051440	052105	042040		
6018	033036	051125	047111	020107		
6019	033044	020101	020062	042523		
6020	033052	052103	051117	053440		
6021	033060	044522	042524	041040		
6022	033066	043505	047111	044516		
6023	033074	043516				
6024	033076	050105	052101	052040	.ASCIZ	<'15'><'12'>AT THE LAST SECTOR OF THE PACK
6025	033104	042510	046040	051501		
6026	033112	020124	042523	052103		
6027	033120	051117	047440	020106		
6028	033126	044124	020105	040520		
6029	033134	045503	000			
6030	033137	047	051105	023522	EM14: .ASCIZ	'ERR' DIDN'T SET WITH 'EOP'
6031	033144	042040	042111	023516		
6032	033152	020124	042523	020124		
6033	033160	044527	044124	023440		
6034	033166	047505	023520	000		
6035	033173	047	047505	023520	EM15: .ASCIZ	'EOP' DIDN'T CLEAR RPCA
6036	033200	042040	042111	023516		
6037	033206	020124	046103	040505		
6038	033214	020122	050122	040503		
6039	033222	000				
6040	033223	123	041525	020101	EM16: .ASCIZ	SUCA NOT CORRECT AFTER 'EOP'
6041	033230	047516	020124	047503		
6042	033236	051122	041505	020124		
6043	033244	043101	042524	020122		
6044	033252	042447	050117	000047		
6045	033260	051105	047522	020122	EM17: .ASCIZ	ERROR READING TEST SECTOR
6046	033266	042522	042101	047111		
6047	033274	020107	042524	052123		
6048	033302	051440	041505	047524		
6049	033310	000122				
6050	033312	047503	052116	047105	EM20: .ASCII	CONTENTS OF FIRST SECTOR ON PACK CHANGED AFTER<'15'><'12'
6051	033320	051524	047440	020106		
6052	033326	044506	051522	020124		
6053	033334	042523	052103	051117		
6054	033342	047440	020116	040520		

ERROR MESSAGES

6055	033350	045503	041440	040510	
6056	033356	045516	041440	040510	
6057	033364	045510	041440	040510	
6058	033372	047506	041522	047111	.ASCIZ @FORCING 'EOP' DURING WRIT@
6059	033378	047506	041522	047111	
6060	033386	047506	041522	047111	
6061	033394	047506	041522	047111	
6062	033394	047506	041522	047111	
6063	033394	047506	041522	047111	EM21: .ASCII @'PROG' ERROR DIDN'T SET WHEN READ COMMAND ISSUED@
6064	033394	047506	041522	047111	
6065	033394	047506	041522	047111	
6066	033394	047506	041522	047111	
6067	033394	047506	041522	047111	
6068	033394	047506	041522	047111	
6069	033394	047506	041522	047111	
6070	033394	047506	041522	047111	
6071	033394	047506	041522	047111	
6072	033394	047506	041522	047111	.ASCIZ <15><12>@WHILE CONTROLLER BUSY@
6073	033394	047506	041522	047111	
6074	033394	047506	041522	047111	
6075	033394	047506	041522	047111	
6076	033394	047506	041522	047111	
6077	033394	047506	041522	047111	EM22: .ASCIZ @'ERR' DIDN'T SET WITH 'PROG'@
6078	033394	047506	041522	047111	
6079	033394	047506	041522	047111	
6080	033394	047506	041522	047111	
6081	033394	047506	041522	047111	
6082	033394	047506	041522	047111	EM23: .ASCIZ @ERROR ATTEMPTING TO MISFORMAT SECTOR @
6083	033394	047506	041522	047111	
6084	033394	047506	041522	047111	
6085	033394	047506	041522	047111	
6086	033394	047506	041522	047111	
6087	033394	047506	041522	047111	
6088	033394	047506	041522	047111	
6089	033394	047506	041522	047111	EM24: .ASCIZ @ERROR VERIFYING THE MISFORMATTED TEST HEADER@
6090	033394	047506	041522	047111	
6091	033394	047506	041522	047111	
6092	033394	047506	041522	047111	
6093	033394	047506	041522	047111	
6094	033394	047506	041522	047111	
6095	033394	047506	041522	047111	
6096	033394	047506	041522	047111	
6097	033394	047506	041522	047111	EM25: .ASCIZ @MISFORMATTED TEST HEADER IS NOT CORRECT@
6098	033394	047506	041522	047111	
6099	033394	047506	041522	047111	
6100	033394	047506	041522	047111	
6101	033394	047506	041522	047111	
6102	033394	047506	041522	047111	
6103	033394	047506	041522	047111	
6104	033394	047506	041522	047111	EM26: .ASCIZ @'HNF' DIDN'T SET WHEN SEARCHING FOR MISFORMATTED SECTOR@
6105	033394	047506	041522	047111	
6106	033394	047506	041522	047111	
6107	033394	047506	041522	047111	
6108	033394	047506	041522	047111	
6109	033394	047506	041522	047111	
6110	033394	047506	041522	047111	

ERROR MESSAGES

6111	034040	040515	052124	042105			
6112	034046	051440	041505	047524			
6113	034054	000122					
6114	034056	042047	045523	020047	EM27:	.ASCIZ	@'DSK' NOT SET WITH 'HNF'@
6115	034064	047516	020124	042523			
6116	034072	020124	044527	044124			
6117	034100	023440	047110	023506			
6118	034106	000					
6119	034107	047					
6120	034114	044504	042510	020047	EM30:	.ASCIZ	@'HE' DIDN'T SET WITH 'HNF'@
6121	034122	051440	047104	052047			
6122	034122	052105	052105	053440			
6123	034130	020110	020110	044047			
6124	034136	000047					
6125	034142	047547	051122	020047	EM31:	.ASCIZ	@'ERR' NOT SET WITH 'HNF'@
6126	034150	047516	020124	042523			
6127	034156	020124	044527	044124			
6128	034164	023440	047110	023506			
6129	034172	000					
6130	034173	05105	051122	051117	EM32:	.ASCIZ	@ERROR WHILE RESTORING MISFORMATTED TEST HEADER@
6131	034200	053440	044510	042514			
6132	034206	051040	051505	047524			
6133	034214	044522	043516	046440			
6134	034222	051511	047506	046522			
6135	034230	052101	042524	020104			
6136	034236	042524	052123	044040			
6137	034244	040505	042504	000122			
6138	034252	047503	052116	047522	EM33:	.ASCII	@CONTROLLER DIDN'T BECOME BUSY WHEN READ COMMAND@
6139	034260	06114	051105	042040			
6140	034266	02111	023516	020124			
6141	034274	042502	047503	042515			
6142	034302	041040	051525	020131			
6143	034310	044127	047105	051040			
6144	034316	040505	020104	047503			
6145	034324	046515	047101	004			
6146	034331	015	044412	051523		.ASCIZ	<15><12>@ISSUED TO SEEKING DRIVE@
6147	034336	042525	020104	047524			
6148	034344	051440	042505	044513			
6149	034352	043516	042040	044522			
6150	034360	042526	000				
6151	034363	105	051122	051117	EM34:	.ASCIZ	@ERROR DURING READ COMMAND WHICH WAS ISSUED TO SEEKING DRIVE@
6152	034370	042040	051125	047111			
6153	034376	020107	042522	042101			
6154	034404	041440	046517	040515			
6155	034412	042116	053440	044510			
6156	034420	044103	053440	051501			
6157	034426	044440	051523	042525			
6158	034434	020104	047524	051440			
6159	034442	042505	044513	043516			
6160	034450	042040	044522	042526			
6161	034456	000					
6162	034457	104	052101	020101	EM35:	.ASCIZ	@DATA INCORRECT FROM HEADER READ COMMAND ISSUED TO A SEEKING DRIVE@
6163	034464	047111	047503	051122			
6164	034472	041505	020124	051106			
6165	034500	046517	044040	040505			
6166	034506	042504	020122	042522			
616E	034514	042101	041440	046517			

ERROR MESSAGES

6167	034522	040515	042116	044440	
6168	034530	051523	042116	020104	
6169	034536	047523	042116	051440	
6170	034544	042116	042116	043516	
6171	034552	042116	042116	042526	
6172	034560	000			
6173	034561	047	054116	042515	EM36: .ASCIZ @'NXME' DIDN'T SET WHEN LOCATION 760000 ADDRESSED@
6174	034566	047	044504	047104	
6175	034574	052047	051440	052105	
6176	034602	053440	042510	020116	
6177	034610	047514	040503	044524	
6178	034616	047117	033440	030066	
6179	034624	030060	020060	042101	
6180	034632	051104	051505	042523	
6181	034640	000104			
6182	034642	044047	023505	042040	EM37: .ASCIZ @'HE' DIDN'T SET WITH 'NXME'@
6183	034650	042111	023516	020124	
6184	034656	042523	020124	044527	
6185	034664	044124	023440	054116	
6186	034672	042515	000047		
6187	034676	042447	051122	020047	EM40: .ASCIZ @'ERR' DIDN'T SET WITH 'NXME'@
6188	034704	044504	047104	052067	
6189	034712	051440	052105	053440	
6190	034720	052111	020110	047047	
6191	034726	046530	023505	000	
6192	034733	0105	051122	051117	EM41: .ASCIZ @ERROR FORMATTING TRACK @
6193	034740	043040	051117	040515	
6194	034746	052124	047111	020107	
6195	034754	051124	041501	020113	
6196	034762	000060			
6197	034764	051105	047522	020122	EM42: .ASCIZ @ERROR READING THE HEADER FROM THE TEST SECTOR@
6198	034772	042522	042101	047111	
6199	035000	020107	044124	020105	
6200	035006	042510	042101	051105	
6201	035014	043040	047522	020115	
6202	035022	044124	020105	042524	
6203	035030	052123	051440	041505	
6204	035036	047524	000122		
6205	035042	042523	052103	051117	EM43: .ASCIZ @SECTOR FIELD FROM HEADER IS NOT CORRECT@
6206	035050	043040	042511	042114	
6207	035056	043040	047522	020115	
6208	035064	042510	042101	051105	
6209	035072	044440	020123	047516	
6210	035100	020124	047503	051122	
6211	035106	041505	000124		
6212	035112	051105	047522	020122	EM44: .ASCIZ @ERROR WRITING SECTOR ADDRESS IN DATA FIELD@
6213	035120	051127	052111	047111	
6214	035126	020107	042523	052103	
6215	035134	051117	040440	042104	
6216	035142	042522	051523	044440	
6217	035150	020116	040504	040524	
6218	035156	043040	042511	042114	
6219	035164	000			
6220	035165	047	047523	023524	EM45: .ASCIZ @'SOT' OR SECTOR ADDRESS REGISTER IS NOT CORRECT@
6221	035172	047440	020122	042523	
6222	035200	052103	051117	040440	

6333	0000	0000	0000	0000	0000
6334	0000	0000	0000	0000	0000
6335	0000	0000	0000	0000	0000
6336	0000	0000	0000	0000	0000
6337	0000	0000	0000	0000	0000
6338	0000	0000	0000	0000	0000
6339	0000	0000	0000	0000	0000
6340	0000	0000	0000	0000	0000
6341	0000	0000	0000	0000	0000
6342	0000	0000	0000	0000	0000
6343	0000	0000	0000	0000	0000
6344	0000	0000	0000	0000	0000
6345	0000	0000	0000	0000	0000
6346	0000	0000	0000	0000	0000
6347	0000	0000	0000	0000	0000
6348	0000	0000	0000	0000	0000
6349	0000	0000	0000	0000	0000
6350	0000	0000	0000	0000	0000
6351	0000	0000	0000	0000	0000
6352	0000	0000	0000	0000	0000
6353	0000	0000	0000	0000	0000
6354	0000	0000	0000	0000	0000
6355	0000	0000	0000	0000	0000
6356	0000	0000	0000	0000	0000
6357	0000	0000	0000	0000	0000
6358	0000	0000	0000	0000	0000
6359	0000	0000	0000	0000	0000
6360	0000	0000	0000	0000	0000
6361	0000	0000	0000	0000	0000
6362	0000	0000	0000	0000	0000
6363	0000	0000	0000	0000	0000
6364	0000	0000	0000	0000	0000
6365	0000	0000	0000	0000	0000
6366	0000	0000	0000	0000	0000
6367	0000	0000	0000	0000	0000
6368	0000	0000	0000	0000	0000
6369	0000	0000	0000	0000	0000
6370	0000	0000	0000	0000	0000
6371	0000	0000	0000	0000	0000
6372	0000	0000	0000	0000	0000
6373	0000	0000	0000	0000	0000
6374	0000	0000	0000	0000	0000
6375	0000	0000	0000	0000	0000
6376	0000	0000	0000	0000	0000
6377	0000	0000	0000	0000	0000
6378	0000	0000	0000	0000	0000
6379	0000	0000	0000	0000	0000
6380	0000	0000	0000	0000	0000
6381	0000	0000	0000	0000	0000
6382	0000	0000	0000	0000	0000
6383	0000	0000	0000	0000	0000
6384	0000	0000	0000	0000	0000
6385	0000	0000	0000	0000	0000
6386	0000	0000	0000	0000	0000
6387	0000	0000	0000	0000	0000
6388	0000	0000	0000	0000	0000
6389	0000	0000	0000	0000	0000
6390	0000	0000	0000	0000	0000

.ASCIZ @2 WORD WRITED

EM61: .ASCIZ @ERROR WRITING TEST SECTOR(S)@

EM62: .ASCIZ @ERROR READING TEST SECTOR(S)@

EM63: .ASCII @DRIVE DIDN'T RETURN TO CYL 0 FROM THE MAXIMUM@'15 '12

.ASCIZ @CYL ON CONTROLLER POWER FAIL@

EM64: .ASCII @CONTENTS OF MEMORY CHANGED DURING POWER@'15 '12

.ASCIZ @FAIL WHILE READING THE DISK@

EM65: .ASCIZ @'SUCL' SET WITH DRIVE DISABLED@

EM66: .ASCIZ @'SUOL' NOT SET WITH DRIVE ENABLED@

ERROR MESSAGES

6391	037074	046102	042105	000
6392	037101	046102	042105	000
6393	037101	046102	042105	000
6394	037101	046102	042105	000
6395	037101	046102	042105	000
6396	037101	046102	042105	000
6397	037101	046102	042105	000
6398	037101	046102	042105	000
6399	037101	046102	042105	000
6400	037101	046102	042105	000
6401	037101	046102	042105	000
6402	037101	046102	042105	000
6403	037101	046102	042105	000
6404	037101	046102	042105	000
6405	037101	046102	042105	000
6406	037101	046102	042105	000
6407	037101	046102	042105	000
6408	037101	046102	042105	000
6409	037101	046102	042105	000
6410	037101	046102	042105	000
6411	037101	046102	042105	000
6412	037101	046102	042105	000
6413	037101	046102	042105	000
6414	037101	046102	042105	000
6415	037101	046102	042105	000
6416	037101	046102	042105	000
6417	037101	046102	042105	000
6418	037101	046102	042105	000
6419	037101	046102	042105	000
6420	037101	046102	042105	000
6421	037101	046102	042105	000
6422	037101	046102	042105	000
6423	037101	046102	042105	000
6424	037101	046102	042105	000
6425	037101	046102	042105	000
6426	037101	046102	042105	000
6427	037101	046102	042105	000
6428	037101	046102	042105	000
6429	037101	046102	042105	000
6430	037101	046102	042105	000
6431	037101	046102	042105	000
6432	037101	046102	042105	000
6433	037101	046102	042105	000
6434	037101	046102	042105	000
6435	037101	046102	042105	000
6436	037101	046102	042105	000
6437	037101	046102	042105	000
6438	037101	046102	042105	000
6439	037101	046102	042105	000
6440	037101	046102	042105	000
6441	037101	046102	042105	000
6442	037101	046102	042105	000
6443	037101	046102	042105	000
6444	037101	046102	042105	000
6445	037101	046102	042105	000
6446	037101	046102	042105	000
6447	037101	046102	042105	000
6448	037101	046102	042105	000
6449	037101	046102	042105	000
6450	037101	046102	042105	000

EM67: .ASCIZ @'SUWP' NOT SET WITH DRIVE SET TO 'READ ONLY'@

EM70: .ASCIZ @'SUWP' SET WITH DRIVE SET TO 'READ WRITE'@

EM71: .ASCII @'SUWP' SET WITH RP11 'WRITE LOCKOUT' SET, CYLINDER@ 15, '12,'

.ASCIZ @LCA'S = 0, AND RPCA = 0@

EM72: .ASCIZ @'SUWP' SET WITH RPCA = 2 & CYL LCA'S = 0@

EM73: .ASCIZ @'SUWP' NOT SET; RPCA = VALUE IN CYL LCA'S@

EM74: .ASCIZ @'SUWP' SET WITH RPCA ONE GREATER THAN CYL LCA VALUES

EM75: .ASCIZ @'SUWP' SET WITH DRIVE LCA SWITCHES EQUAL TO SELECTED DRIVES

ERROR MESSAGES

6496	00000000	00000000	00000000	00000000	EM76: .ASCIZ @DRIVE HAS GONE OFFLINE@
6497	00000000	00000000	00000000	00000000	EM77: .ASCIZ @DRIVE IS UNSAFE@
6498	00000000	00000000	00000000	00000000	EM100: .ASCIZ @DRIVE TIMED OUT@
6499	00000000	00000000	00000000	00000000	EM101: .ASCIZ @CONTROLLER TIMED OUT@
6500	00000000	00000000	00000000	00000000	EM102: .ASCIZ @DATA COMPARE ERROR@
6501	00000000	00000000	00000000	00000000	EM103: .ASCIZ @DATA COMPARE ERROR (MEMORY MANAGEMENT ENABLED,@
6502	00000000	00000000	00000000	00000000	EM106: .ASCIZ @'ERR' DIDN'T SET WITH 'WCE'@
6503	00000000	00000000	00000000	00000000	EM107: .ASCIZ @'HE' DIDN'T SET WITH 'PROG'@
6504	00000000	00000000	00000000	00000000	EM112: .ASCIZ @DRIVE UNSAFE AFTER POWER FAIL@
6505	00000000	00000000	00000000	00000000	EM113: .ASCIZ @'SUMP' NOT SET WITH DRIVE L0A SWITCHES GREATER THAN SELECTED DRIVES

6503	010272	010510	020116	042523
6504	010300	010514	020103	042505
6505	010306	010540	044522	042526
6506	010314	000		
6508				
6509				
6510	010303	122	040520	051104
6511	010303	000		
6512	010303	000	051505	020124
6513	010303	000	022140	051122
6514	010303	000	020103	042040
6515	010303	000	021440	020040
6516	010303	000	021440	020123
6517	010303	000	051040	020040
6518	010303	000	020040	051040
6519	010303	000	020040	020040
6520	010303	000	053520	000103
6521	010303	000	040502	020040
6522	010303	000	050122	040503
6523	010303	000	020040	050122
6524	010303	000	020040	020040
6525	010303	000	030515	020040
6526	010303	000	052523	040503
6527	010303	000	020040	044523
6528	010303	000	000	
6529	010303	124	051505	020124
6530	010303	000	022140	051122
6531	010303	000	020103	042040
6532	010303	000	021440	020040
6533	010303	000	052101	047122
6534	010303	000	051040	042120
6535	010303	000	020040	051040
6536	010303	000	020122	020040
6537	010303	000	041520	000123
6538	010303	000	041527	020040
6539	010303	000	050122	040502
6540	010303	000	020040	050122
6541	010303	000	020040	020040
6542	010303	000	040504	020040
6543	010610	000	050122	030515
6544	010616	000	020040	052523
6545	010621	000	040503	020040
6546	010622	000	047514	000
6547	010637	124	051505	020124
6548	010644	000	042440	051122
6549	010652	000	020103	042040
6550	010660	000	021440	020040
6551	010666	000	050130	023524
6552	010674	000	051461	020124
6553	010702	000	042122	031040
6554	010710	000	053440	051117
6555	010716	000		
6556	010720	000	052123	021440
6557	010726	000	051105	020122
6558	010734	01520	020040	051104

: 'DH' MESSAGE LINES

DH1: .ASCIZ @RPADR@

DH2: .ASCIZ @TEST @ ERR PC DRV @ RPDS RPER RPDS RPWC@

DH2A: .ASCIZ @RPBA RPBA RPDA RPM1 SUCA SILOS

DH5: .ASCIZ @TEST @ ERR PC DRV @ PATRN @ RPDS RPER RPDS@

DH5A: .ASCIZ @RPWC RPBA RPBA RPDA RPM1 SUCA SILOS

DH11: .ASCIZ @TEST @ ERR PC DRV @ EXPT'D 1ST WORD 2ND WORD@

DH12: .ASCIZ @TEST @ ERR PC DRV @ EXPT'D RECV'D LOGNS

ERROR MESSAGES

```

6615 020040 051040 041120
6616 020101 020040 051040
6617 020101 020040 020040
6618 051040 051040 000101
6619 042510 042510 051105
6620 020101 042524 042524
6621 020040 020040 020040
6622 020040 020040 020040
6623 020040 020040 020040
6624 020040 020040 020040
6625 020040 020040 020040
6626 020040 020040 020040
6627 020040 020040 020040
6628 020040 020040 042517
6629 020040 020040 042523
6630 042524 052123 005015
6631 020040 051105 020122
6632 020040 020040 051104
6633 020126 020043 020040
6634 040526 052514 020105
6635 020040 052517 052116
6636 051105 020040 042101
6637 051104 000000
6638 020122 042120 020123
6639 020040 051040 042520
6640 020122 020040 051040
6641 041520 020123 020040
6642 020040 042120 000101
6643 020040 020040 020040
6644 020040 020040 020040
6645 020040 020040 020040
6646 020040 020040 020040
6647 020040 020040 020040
6648 042047 020040 042522
6649 053103 042047 005015
6650 042524 052123 021440
6651 020040 051105 020122
6652 041520 020040 051104
6653 020126 020043 020040
6654 042523 052103 051117
6655 020040 040504 040524
6656 020040 020040 040504
6657 040524 000000
6658 051505 020124 020124
6659 020043 042440 051122
6660 050040 020103 042040
6661 053122 021440 020040
6662 042440 050130 023524
6663 020104 051040 041505
6664 020040 050122 020040
6665 050122 051504 020040
6666 020040 050122 051105
6667 020040 020040 050122
6668 051503 020040 020040
6669 050122 020040 020040
6670 050122 040503 020040

```

DH43B: .ASCIZ @HEADER CONTENTS@

DH45: .ASCII @ EXPT'D 'SOT' SECTOR@ 15' 12'

.ASCIZ @TEST @ ERR PC DRV @ VALUE COUNTER ADDR@

DH45A: .ASCIZ @RPOS RPER RPCS RPD@

DH46: .ASCII @ EXPT'D RECV'D@ 15' 12'

.ASCIZ @TEST @ ERR PC DRV @ SECTOR DATA DATA@

DH47: .ASCIZ @TEST @ ERR PC DRV @ EXPT D RECV'D@

DH47A: .ASCIZ @RPOS RPER RPCS RPD@

6671	020040	050122	040504										
6672	0000												
6673	0000	051505	020124	DH51:	.ASCIZ	QTEST	ERR PC	DRV	RPDS	RPER	RPCS	RPWC	RPBQ
6674	0000	051505	051122										
6675	0000	051505	051122										
6676	0000	051505	051122										
6677	0000	051505	051122										
6678	0000	051505	051122										
6679	0000	051505	051122										
6680	0000	051505	051122										
6681	0000	051505	051122										
6682	0000	051505	051122										
6683	0000	051505	051122										
6684	0000	051505	051122										
6685	0000	051505	051122										
6686	0000	051505	051122										
6687	0000	051505	051122										
6688	0000	051505	051122										
6689	0000	051505	051122										
6690	0000	051505	051122										
6691	0000	051505	051122										
6692	0000	051505	051122										
6693	0000	051505	051122										
6694	0000	051505	051122										
6695	0000	051505	051122										
6696	0000	051505	051122										
6697	0000	051505	051122										
6698	0000	051505	051122										
6699	0000	051505	051122										
6700	0000	051505	051122										
6701	0000	051505	051122										
6702	0000	051505	051122										
6703	0000	051505	051122										
6704	0000	051505	051122										
6705	0000	051505	051122										
6706	0000	051505	051122										
6707	0000	051505	051122										
6708	0000	051505	051122										
6709	0000	051505	051122										
6710	0000	051505	051122										
6711	0000	051505	051122										
6712	0000	051505	051122										
6713	0000	051505	051122										
6714	0000	051505	051122										
6715	0000	051505	051122										
6716	006504	012	040502										
6717	0101	042104	020122		.ASCIZ	QADDR	DATA	DATA					
6718	020040	042040	052101										
6719	020101	020040	042040										
6720	052101	000101											
6721	044513	040520	031522	DH103:	.ASCIZ	QKIPAR3	KIPAR4	KIPARS	KIPDR3	KIPDR4	KIPDR5		
6722	020040	044513	040520										
6723	032122	020040	044513										
6724	040520	032522	020040										
6725	044513	042120	031522										
6726	020040	044513	042120										

ERROR MESSAGES

6727	0432122	0020040	044513		
6728	0432122	0020040	044513		
6729	0432122	0020040	044513		
6730	0432122	0020040	044513		
6731	0432122	0020040	044513		
6732	0432122	0020040	044513		
6733	0432122	0020040	044513		
6734	0432122	0020040	044513		
6735	0432122	0020040	044513		
6736	0432122	0020040	044513		
6737	0432122	0020040	044513		
6738	0432122	0020040	044513		
6739	0432122	0020040	044513		
6740	0432122	0020040	044513		
6741	0432122	0020040	044513		
6742	0432122	0020040	044513		
6743	0432122	0020040	044513		
6744	0432122	0020040	044513		
6745	0432122	0020040	044513		
6746	0432122	0020040	044513		
6747	0432122	0020040	044513		
6748	0432122	0020040	044513		
6749	0432122	0020040	044513		
6750	0432122	0020040	044513		
6751	0432122	0020040	044513		
6752	0432122	0020040	044513		
6753	0432122	0020040	044513		
6754	0432122	0020040	044513		
6755	0432122	0020040	044513		
6756	0432122	0020040	044513		
6757	0432122	0020040	044513		
6758	0432122	0020040	044513		
6759	0432122	0020040	044513		
6760	0432122	0020040	044513		
6761	0432122	0020040	044513		
6762	0432122	0020040	044513		
6763	0432122	0020040	044513		
6764	0432122	0020040	044513		
6765	0432122	0020040	044513		
6766	0432122	0020040	044513		
6767	0432122	0020040	044513		
6768	0432122	0020040	044513		
6769	0432122	0020040	044513		
6770	0432122	0020040	044513		
6771	0432122	0020040	044513		
6772	0432122	0020040	044513		
6773	0432122	0020040	044513		
6774	0432122	0020040	044513		
6775	0432122	0020040	044513		
6776	0432122	0020040	044513		
6777	0432122	0020040	044513		
6778	0432122	0020040	044513		
6779	0432122	0020040	044513		
6780	0432122	0020040	044513		
6781	0432122	0020040	044513		
6782	0432122	0020040	044513		

DM103A: .ASCII @VIRTUAL GOOD BAD@<15><12>

.ASCIZ @ADDR DATA DATA@

DM105: .ASCIZ @TOTAL COMPARE ERRORS:@

DM110: .ASCIZ @UNSUCCESSFUL AFTER 3 RETRIES@

DM111: .ASCII @RETRY SUCESSFUL@<15><12>

.ASCIZ @RETRY COUNT:@

.EVEN

: 'DT' WORDS

DT1: .WORD RPADR
DT2: .WORD \$TMPD, \$ERRPC, CHKDRV, \$RPDS, \$RPER, \$RPCS, \$RPWC

.WORD \$RPBA, \$RPCA, \$RPDA, \$RPM1, \$SUCA, \$SILO

DT5: .WORD \$TMPD, \$ERRPC, CHKDRV, PATNUM, \$RPDS, \$RPER, \$RPCS

.WORD \$RPWC, \$RPBA, \$RPCA, \$RPDA, \$RPM1, \$SUCA, \$SILO

DT11: .WORD \$TMPD, \$ERRPC, CHKDRV, \$GDDAT, BUFFER, BUFFER+2

DT12: .WORD \$TMPD, \$ERRPC, CHKDRV, \$GDDAT, \$BDDAT, \$BDADR

DT14: .WORD \$TMPD, \$ERRPC, CHKDRV, \$RPDS, \$RPER, \$RPCS

DT16: .WORD \$TMPD, \$ERRPC, CHKDRV, MAXCYL, \$SUCA

DT25: .WORD \$TMPD, \$ERRPC, CHKDRV, \$GDDAT, \$BDDAT

DT43: .WORD BUFFER, BUFFER+2, BUFFER+4
.WORD \$TMPD, \$ERRPC, CHKDRV, \$GDDAT, BUFFER+4

.WORD \$RPDS, \$RPER, \$RPCS, \$RPWC, \$RPBA, \$RPCA, \$RPDA

6783	043244	001342	001344	001346			
6784	043252	001350					
6785	043254	043656	043660	043662			
6786	043272	001116	001116	001176	DT45:	.WORD	BUFFER, BUFFER+2, BUFFER+4
6787	043270	001124	001126	001122		.WORD	STMPD, \$ERRPC, CHKDRV, \$GDDAT, \$BDDAT, \$BDADR
6788	043276	001334	001336	001340		.WORD	\$RPDS, \$RPER, \$RPCS, \$RPDA
6789	043304	001350					
6790	043306	001160	001116	001176	DT46:	.WORD	STMPD, \$ERRPC, CHKDRV, \$GDADR, \$GDDAT, \$BDDAT
6791	043314	001120	001124	001126			
6792	043322	001160	001116	001176	DT47:	.WORD	STMPD, \$ERRPC, CHKDRV, \$GDDAT, BUFFER
6793	043330	001124	043656				
6794	043334	001334	001336	001340		.WORD	\$RPDS, \$RPER, \$RPCS, \$RPCA, \$RPDA
6795	043342	001346	001350				
6796	043346	001160	001116	001176	DT51:	.WORD	STMPD, \$ERRPC, CHKDRV, \$RPDS, \$RPER, \$RPCS, \$RPWC, \$RPBA
6797	043354	001334	001336	001340			
6798	043362	001342	001344				
6799	043366	001160	001116	001176	DT63:	.WORD	STMPD, \$ERRPC, CHKDRV, \$RPDS, \$RPER, \$RPCS, \$SUCA
6800	043374	001334	001336	001340			
6801	043402	001354					
6802	043404	001160	001116	001176	DT71:	.WORD	STMPD, \$ERRPC, CHKDRV, \$RPDS, \$RPER, \$RPCS, \$RPCA
6803	043412	001334	001336	001340			
6804	043420	001346					
6805	043422	001160	001116	001176	DT102:	.WORD	STMPD, \$ERRPC, CHKDRV, CMBUF, CMCYL, CMTRK, CMSEC, PATNUM
6806	043430	025430	025422	025426			
6807	043436	025424	001204				
6808	043442	001122	001124	001126		.WORD	\$BDADR, \$GDDAT, \$BDDAT
6809	043450	001160	001116	001176	DT103:	.WORD	STMPD, \$ERRPC, CHKDRV, CMBUF, CMCYL, CMTRK, CMSEC, PATNUM
6810	043456	025430	025422	025426			
6811	043464	025424	001204				
6812	043470	172346	172350	172352		.WORD	KIPAR3, KIPAR4, KIPARS, KIPDR3, KIPDR4, KIPDRS
6813	043476	172306	172310	172312			
6814	043504	001122	001124	001126	DT104:	.WORD	\$BDADR, \$GDDAT, \$BDDAT
6815	043512	025414			DT105:	.WORD	ERCTR
6816	043514	001314			DT111:	.WORD	RETRY
6817							
6818							
6819							
6820	043516	000001					
6821	043520	001			DF1:	.WORD	1
6822	043521	000				.BYTE	1
6823						.BYTE	0
6824	043522	000002					
6825	043524	007			DF2:	.WORD	2
6826	043525	000				.BYTE	7
6827	043526	040410				.BYTE	0
6828	043530	006				.WORD	DH2A
6829	043531	000				.BYTE	6
6830						.BYTE	0
6831	043532	000002					
6832	043534	007			DF5:	.WORD	2
6833	043535	010				.BYTE	7
6834	043536	040552				.BYTE	10
6835	043540	007				.WORD	DH5A
6836	043541	000				.BYTE	7
6837						.BYTE	0
6838	043542	000001			DF11:	.WORD	1

ERROR MESSAGES

6839	043544	006		.BYTE	6
6840	043545	000		.BYTE	0
6841					
6842	043546	000002	DF25:	.WORD	2
6843	043550	005		.BYTE	0
6844	043551	000		.BYTE	0
6845	043552	041236		.WORD	0H25A
6846	043553	003		.BYTE	0
6847	043555	000		.BYTE	0
6848					
6849	043556	000003	DF43:	.WORD	3
6850	043560	006		.BYTE	0
6851	043561	000		.BYTE	0
6852	043562	041375		.WORD	0H43A
6853	043564	005		.BYTE	0
6854	043565	000		.BYTE	0
6855	043566	041462		.WORD	0H43B
6856	043570	003		.BYTE	0
6857	043571	000		.BYTE	0
6858					
6859	043572	000002	DF45:	.WORD	2
6860	043574	006		.BYTE	0
6861	043575	000		.BYTE	0
6862	043576	041637		.WORD	0H45A
6863	043600	004		.BYTE	4
6864	043601	000		.BYTE	0
6865					
6866	043602	000002	DF47:	.WORD	2
6867	043604	005		.BYTE	0
6868	043605	000		.BYTE	0
6869	043606	042100		.WORD	0H47A
6870	043610	005		.BYTE	0
6871	043611	000		.BYTE	0
6872					
6873	043612	000001	DF51:	.WORD	1
6874	043614	010		.BYTE	8
6875	043615	000		.BYTE	0
6876					
6877	043616	000001	DF53:	.WORD	1
6878	043620	005		.BYTE	5
6879	043621	000		.BYTE	0
6880					
6881	043622	000001	DF63:	.WORD	1
6882	043624	007		.BYTE	7
6883	043625	000		.BYTE	0
6884					
6885	043626	000002	DF102:	.WORD	2
6886	043630	010		.BYTE	8
6887	043631	200		.BYTE	200
6888	043632	042514		.WORD	0H102A
6889	043634	000		.BYTE	0
6890	043635	000		.BYTE	0
6891					
6892	043636	000003	DF103:	.WORD	3
6893	043640	010		.BYTE	8
6894	043641	200		.BYTE	200

ERROR MESSAGES

```

6895 043642 042566
6896 043644 006
6897 043645 000
6898 043646 042645
6900 043650 000
6901 043651 000
6902 043652 000001
6903 043654 003
6904 043655 000
6905
6906
6907
6908
6909
6910 043656
6911 043656 005015 046412 044501
6912 043664 042116 041505 030455
6913 043672 026461 055104 050122
6914 043700 026531 103
6915 043703 015 051012 030520
6916 043710 026461 020105 052506
6917 043716 041516 044524 047117
6918 043724 046101 046040 043517
6919 043732 041511 023040 051040
6920 043740 040505 027504 051127
6921 043746 052111 020105 042524
6922 043754 052123 005015 000012
6923 000001

```

```

.WORD DM103
.BYTE 6
.BYTE 0
.WORD DM103A
.BYTE 0
.BYTE 0
DF104: .WORD 1
.BYTE 3
.BYTE 0

```

.EVEN

BUFFER=. ;BUFFER STARTS HERE

TITLE: .ASCII <15><12><12>@MAINDEC-11-DZRPY-C@

.ASCIIZ <15><12>@RP11-E FUNCTIONAL LOGIC & READ/WRITE TEST@<15><12><12>

.END

SYMBOL TABLE

\$SCORE	022402	\$GDOAT	001124	\$OVER	022062	\$SIZE	022436	\$TRAP	022502
\$SCRLF	001173	\$GET42	017040	\$PASS	001100	\$STUP	= 177777	\$TRAP2	022524
\$SCROUT	022432	\$GTSMR	020774	\$PQUIS	001172	\$SUCA	001354	\$TRK	002513
\$SCYL	002510	\$HO	= 000000	\$ROCHR	021246	\$SVLAD	022034	\$TRP	= 000014
\$SOBK	017036	\$ICNT	001104	\$ROLIN	021336	\$SVPC	= 000200	\$TRPAD	= 022536
\$SOBGN	017060	\$INTAG	001135	\$ROSRM	= 000012	\$SWR	167000	\$TSTNM	001102
\$SOTBL	020356	\$ITEMB	001114	\$RTSARM	= 022136	\$SWRKM	= 000000	\$TTYIN	021572
\$SNOAO	017050	\$KTINEX	022374	\$RTPCD	001344	\$TIMES	001162	\$TYPOS	020152
\$SNOCT	017004	\$KTOUT	022234	\$RTPCD	001344	\$TKB	001146	\$TYPE	017504
\$SNULL	017064	\$KT11	022232	\$RTPCD	001344	\$TKCNT	020376	\$TYPEC	017654
\$SOP	016616	\$L	001174	\$RTPCD	001344	\$TKINT	020416	\$TYPEX	017722
\$SOPCT	016776	\$LPROP	001106	\$RTPCD	001344	\$TKOEN	= 020416	\$TYPOC	017750
\$SPLG	001103	\$LPERR	001110	\$RTPCD	001344	\$TKOIN	020400	\$TYPON	017764
\$SPLX	001115	\$LSTRO	022476	\$RTPCD	001344	\$TKOOU	020402	\$TYPOS	017764
\$SRROR	017070	\$LSTBK	022500	\$RPLC	001350	\$TKSR	020404	\$W	002504
\$SRRPC	001116	\$MNEW	021634	\$RTNAD	017062	\$TKS	001144	\$XTSTR	021660
\$SRRTB	002514	\$MSUR	021623	\$SAVRD	022100	\$TKSRV	020466	\$SET4	= 000000
\$SRTTL	001112	\$MXCNT	022076	\$SCOPT	021646	\$TMPD	001160	\$FILL	= 020147
\$RECAP	001164	\$NULL	001154	\$SEC	002512	\$TN	= 000020		= 043762
\$FILLC	001156	\$NLTST	= 000001	\$SETUP	= 000122	\$TPB	001152		
\$FILLS	001155	\$SOCNT	020146	\$SILO	001356	\$TPFLG	001157		
\$GORDR	001120	\$SOMODE	020150	\$SIZE	022174	\$TPS	001150		

. ABS. 043762 000

ERRORS DETECTED: 0

DSKW:DZRPYC DSKW:DZRPYC SOL=DSKW:SYSMAC.SML,DSKM:DZRPYC.P11
 RUN-TIME: 16 24 .6 SECCNDS
 RUN-TIME RATIO: 371 41=8.9
 CORE USED: 37K 23 PAGES.

E11