

PDP11/70

PDP11/70 MEMORY TEST
MD-11-DEMJA-C

EP-DEMJA-C-DL-A
COPYRIGHT © 73-77
FICHE 1 OF 1

AUG 1977
digital
MADE IN USA

Table with multiple columns and rows of data, likely representing memory test results. The content is extremely faint and illegible due to the low contrast of the scan. It appears to be a structured list or grid of information.

B01

EJF1DZDRGSEQ
PDP10 411

00010000

770720

PDP10 411

(H0R1DEMJACSEQ

00010000

770720

IDENTIFICATION

PRODUCT CODE: MAINDEC-11-DEMJA-C-D
PRODUCT NAME: PDP-11/70 MEMORY TEST
DATE CREATED: MAY 1977
MAINTAINER: DIAGNOSTIC RELEASE ENGINEERING
AUTHOR: JIM LACEY

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

THE SOFTWARE DESCRIBED IN THIS DOCUMENT IS FURNISHED UNDER A LICENSE AND MAY ONLY BE USED OR COPIED IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE.

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

COPYRIGHT (C) 1973 1977 BY DIGITAL EQUIPMENT CORPORATION

TABLE OF CONTENTS

1.0	Abstract	9.0	Program Description
2.0	Requirements		
	2.1	Equipment	
	2.2	Storage	
	2.3	Preliminary Programs	
3.0	Loading & Starting Procedure		
3.1	ACT11 OPERATION		
4.0	SWITCH SETTINGS		
5.0	SUBROUTINE ABSTRACTS		
	5.1	SCOPE	
6.0	ERRORS		
	6.1	PARITY ERROR	
7.0	RESTRICTIONS		
	7.1	STARTING RESTRICTION	
	7.2	OPERATION RESTRICTION	
8.0	MISCELLANEOUS		
	8.1	STACK POINTER	
	8.2	PASS COUNT	
	8.3	ERROR COUNT	
	8.4	DISPLAY REGISTER	
	8.5	POWER FAIL	
	8.6	EXECUTION TIME	

1.0 ABSTRACT

PROGRAM DEMJA TESTS CONTIGUOUS MEMORY ADDRESS FROM 000000 TO 17757776. IT VERIFIES THAT EACH ADDRESS IS UNIQUE (AN ADDRESS TEST) AND THAT EACH MEMORY LOCATION CAN BE READ/WRTTEN RELIABLY (WORST CASE NOISE TESTS). THIS PROGRAM MAY BE USED TO ADJUST/MARGIN MEMORY.

2.0 REQUIREMENTS

2.1 EQUIPMENT

PDP-11/70 FAMILY PROCESSOR WITH 32K MEMORY

2.2 STORAGE

PROGRAM STORAGE - THE PROGRAM USES MEMORY 0-17777

2.3 PRELIMINARY PROGRAMS

DEKBA THROUGH DEKBF

3.0 LOADING AND STARTING PROCEDURE

LOAD PROGRAM INTO MEMORY USING ABS LOADER
LOAD ADDRESS 200

SET SW12 IN DESIRED POSITION (SEE SEC 4.0)

PRESS START.

ASTERISK "*" WILL BE PRINTED AFTER EACH PASS.

"DEMJA DONE!" WILL BE PRINTED AFTER 6 PASSES.

PASS COUNT MAY BE MONITORED IN THE DISPLAY REGISTER.
NOTE: THIS PROGRAM SAVES THE LOADERS (BOOT AND ABS), TO RESTORE THE LOADERS, RESTART AT 162.

3.1 ACT11 OPERATION

IF THE PROGRAM IS RUN IN QUICK VERIFY MODE UNDER ACT11 THE PROGRAM IS DONE AFTER THE FIRST PASS.

4.0 SWITCH SETTINGS

SW15 = 1 OR UP.... HALT ON ERROR

NOTE: IF SW15=1 WHEN AN ERROR OCCURS THE PROGRAM WILL HALT, AND THE CORRECT DATA WILL NOT BE LOADED INTO THE FAILING ADDRESS. IF SW15 IS RAISED AFTER THE ERROR TYPEOUT BEGINS THE PROGRAM WILL HALT WHEN THE TYPEOUT COMPLETES, AND THE CORRECT DATA WILL BE LOADED INTO THE FAILING ADDRESS.

SW14 = 1 OR UP.... LOOP SUBTEST

SW13 = 1 OR UP..... INHIBIT ERROR TYPEOUT

SW12 = 1 OR UP....INHIBIT USE OF MEMORY MANAGEMENT

NOTE: INHIBITING THE USE OF MEMORY MANAGEMENT CAN BE DONE ONLY WHEN THE PROGRAM IS STARTED.
IF THE USE OF MEMORY MANAGEMENT IS INHIBITED THE LAST

ADDRESS AS TYPED BY THE PROGRAM WILL ONLY REFLECT THE AMOUNT OF MEMORY UP TO 28K (LAST ADDRESS = 160000).

SW11 = 1 OR UP..... INHIBIT SUBTEST ITERATION

SW10 = 1 OR UP..... RING BELL ON ERROR

SW9 = 1 OR UP..... DISPLAY ERROR COUNT IN DISPLAY REGISTER

SW9 = 0 OR DOWN... DISPLAY PASS COUNT IN DISPLAY REGISTER

SW8 = 1 OR UP..... HALT PROGRAM UNRELOCATED & RESTORE LOADERS.

5.0 Subroutine Abstracts

5.1 Scope

the program stores in R1 the PC of the last test successfully executed and may be used as an aid in debugging if the program 'bombs' because of a hardware failure.

6.0 Errors

These tests print out the pc where the error was detected, the failing address, the good data, and the bad data i.e.

PC=xxxxxx ADDRESS aaaaaa GOOD DATA gggggg BAD DATA bbbbbb

the address of the failing location is the true 22 bit physical address.

Note: When testing memory locations 0-77776 the PC typed will be a multiple of 100000 greater than reflected in the program listing

the address of the bad data is in (R2) -2

the good data in R0

the bad data in R3

The address of good data is in R4 (Random Data Test only) when an error is detected when exercising the memory using the worst case noise patterns, the user should restart the program selecting program #2 (see sec 9.1 for details) selecting the appropriate parameters. The user can use the PC and address of the failure to select the proper core bank(s) affected and also the specific pattern. This allows maximum scope capabilities.

6.1 Parity Error

If a parity error is detected the program will type:

PARITY ERROR

PC=PPPPPP MEMORY ADDRESS IS AAAAAAAA

PARITY ERROR REG=EEEEEE ?????????? MARGIN

Where P P P P P P is the contents of the PC when the parity error occurred, A A A A A A A A is the address of the word, E E E E E E is the contents of the memory error register, and ? ? ? ? ? ? ? ? is the margin setting at the time of the parity error.

After reporting the parity error the program will start over.

7.0 Restrictions

7.1 Starting Restriction

Program must not be relocated when restarting

7.2 Operational Restriction

Program checks contiguous memory if a parity error trap occurs when the program is relocated program action is undefined. If parity memory is available or selected the 3xor9 test pattern is for parity memory only. Do not power fail the program when the program is running relocated.

8.0 Miscellaneous

If the program halts in the trap/interrupt vector area (0-1000), examine register 6 (the stack ptr). R6 contains the address where the PC of the instruction that caused the trap abort is stored. See also R1 (R1 specifies the last test completed).

Note: the PDP-11/70 will display the trap vector address+4 in the address lights. Thus a trap to 4 (bus error) will display 10 in the address lights.

8.1 Stack Pointer

The stack pointer is initially set to 520 and is reset to this value at the start of each subtest.

8.2 Pass Count

Six passes are required for completion of this program; at which time an "*" will be printed. the pass count may be observed by turning the switch to the display position. (the pass count is also stored in location 1000.) the pass count should be monitored in the event that the program enters an undefined loop..blank 1

8.3 Error Count

Each time an error occurs, the error count is incremented. The error count can be observed by turning the switch to the display position and setting switch 9. (the error count is also stored in location 1002.) the program will count 17777(8) errors; the error count is not incremented past this value..blank 1

8.4 Display Register

Either the pass count or the error count is displayed in the display register. the count to be displayed is controlled by the setting of switch 9..blank 1

8.5 Power Fail

The program may be power failed when running. When the power returns the program will continue in sequence. ****caution**** do not turn power off/on until the message 'power failed' has been typed. this is because the stack may overflow.

8.5 Execution Time

Execution time is dependent on the amount of memory.

9.0 Program Description

The program verifies each address by writing the value of each address into itself starting at location 20000 and ending at the last location in memory. The value of the last location +2 is typed on the TTY. Next the values written are verified. to complete the address test the complement value of each memory address is written starting at the last memory address and ending at address 20000. The written complement values are then verified. The next phase of testing includes reading, writing and checking memory using worst case noise test pattern. A subtest is dedicated to checking the pattern. The test proceeds by exercising each bank of memory using the worst case pattern. **THE PROGRAM THEN CHECKS MEMORY USING RANDOM DATA (RANTST).** This routine moves the program code throughout memory starting at location 20000, and relocates the data by a 32(10) word offset on each subsequent relocation. i.e., First relocation is to 20000, next is to 20100, then 20200, etc. After relocation the code moved is checked against the original code (0-17776). When the random data test is complete the program then successively rotates a 0 bit (ROTO) and a '1' bit (ROT1) through all of memory. When all testing is complete the program increments the pass count (location 1000) and restarts beginning with the worst case noise tests. An asterisk (*) will be typed on completion of each pass, and when 6 passes have been completed the program will type 'DEMJA DONE' and restart the program beginning with the memory address tests.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56

```

.NLIST MD,MC
.LIST ME
.ABS
.TITLE MAINDEC-11-DEMJA-C PDP11/70 MEMORY TEST
.SBTTL STARTING INST & DEFINITIONS
; COPYRIGHT 1973 1977 DIGITAL EQUIPMENT CORP., MAYNARD, MASS.

; THIS TEST CHECKS THAT ALL MEMORY ADDRESSES ARE UNIQUE USING ADDRESS TESTS
; AND CHECKS DATA RELIABILITY OF MEMORY USING WORST CASE NOISE TEST PATTERNS
; A RANDOM * PATTERN (PROGRAM CODE RELOCATED), A ROTATING 0 AND ROTATING
; 1 PATTERN.

; LOADING AND STARTING INSTRUCTIONS
; LOAD ADDRESS 200 AND START
; THIS PROGRAM ALSO RELOCATES THE ABS AND BOOT LOADERS TO ALLOW TESTING
; OF MEMORY, TO RESTORE THE LOADERS RESTART AT 162.
; STACK POINTER IS SET AT 500
; AN ASTERISK '*' WILL BE PRINTED ON COMPLETION OF EACH PASS, AND
; THE PROGRAM NAME WILL BE PRINTED WHEN TEST IS COMPLETE.

```

; GENERAL REGISTER ASSIGNMENTS

```

000000 R0=%0
000001 R1=%1
000002 R2=%2
000003 R3=%3
000004 R4=%4
000005 R5=%5
000006 SP=%6
000007 PC=%7
000000 R10=%0
000001 R11=%1
000002 R12=%2
000003 R13=%3
000004 R14=%4
000005 R15=%5

```

; STATUS REGISTER (PSW) BIT ASSIGNMENTS

```

000001 C=1 ; C BIT
000002 V=2 ; V BIT
000004 Z=4 ; Z BIT
000010 N=10 ; N BIT
000020 T=20 ; 'T' BIT
000340 PRTY7=340 ; PRIORITY LEVEL 7
000200 PRTY4=200 ; PRIORITY LEVEL 4
000000 KM=000000 ; KERNEL MODE
040000 SM=040000 ; SUPERVISORY MODE
140000 UM=140000 ; USER MODE
000000 PKM=000000 ; PREVIOUS KERNEL MODE
010000 PSM=010000 ; PREVIOUS SUPERVISORY MODE
030000 PUM=030000 ; PREVIOUS USER MODE
004000 REG=004000 ; SELECT R10-R15

```

; VECTOR ADDRESSES

```

000004 ERRVEC=4 ; ADDRESS OF ERROR VECTOR
000010 RESVEC=10 ; ADDRESS OF RESERVED INST. TRAP VECTOR
000014 TBITVEC=14 ; ADDRESS OF 'T' BIT TRAP VECTOR

```



```

57      000014      TRTVEC=14      ;ADDRESS OF 'TRACE' TRAP VECTOR
58      000014      BPTVEC=14      ;ADDRESS OF 'BREAKPOINT' TRAP VECTOR
59      000020      TOTVEC=20      ;ADDRESS OF TOT TRAP VECTOR
60      000024      PFVEC=24      ;ADDRESS OF POWER FAIL TRAP VECTOR
61      000030      EMTVEC=30      ;ADDRESS OF EMT VECTOR
62      000034      TRAPVEC=34      ;ADDRESS OF TRAP VECTOR
63      000060      TKVEC=60      ;ADDRESS OF TTY KEYBOARD INTERRUPT VECTOR
64      000064      TPVEC=64      ;ADDRESS OF TTY PRINTER INTERRUPT VECTOR
65      000240      PIRVEC=240      ;ADDRESS OF PIRQ VECTOR
66      000244      FPEVEC=244      ;ADDRESS OF FLOATING POINT INT. VECTOR
67      000250      MMVEC=250      ;ADDRESS OF MEM MGMT ERROR TRAP VECTOR
68
69      ;REGISTER ADDRESSES
70      177776      PSW=177776      ;ADDRESS OF STATUS REGISTER
71      177774      SLR=177774      ;ADDRESS OF STACK LIMIT REGISTER
72      177772      PIRQ=177772      ;ADDRESS OF PROGRAM INTERRUPT REQUEST
73      177770      UBREAK=177770      ;ADDRESS OF MICRO BREAK REGISTER
74      177746      CNTRL=177746      ;ADDRESS OF 11/70 MEMORY CONTROL REGISTER
75      177560      TKS=177560      ;ADDRESS OF KEYBOARD CSR
76      177562      TKB=177562      ;ADDRESS OF KEYBOARD BUFFER
77      177564      TPS=177564      ;ADDRESS OF TELEPRINTER CSR
78      177566      TPB=177566      ;ADDRESS OF TELEPRINTER BUFFER
79      177570      SWR=177570      ;ADDRESS OF CONSOL SWITCH REGISTER
80      177570      DISPLAY=177570      ;ADDRESS OF CONSOL DISPLAY REGISTER
81
82      ;INITIAL STACK POINTER SETTING
83      000500      STKPTR=500
84
85      ;MISCELLANEOUS BIT ASSIGNMENTS
86      000100      BIT15= 100
87      040000      BIT14= 040000
88      020000      BIT13= 020000
89      010000      BIT12= 010000
90      001000      BIT9= 001000
91      000400      BIT8= 000400
92      000100      BIT6= 000100
93
94      ;MEMORY MANAGEMENT REGISTER ADDRESS ASSIGNMENTS
95      177572      SRO=177572      ;ADDRESS OF MEM MGMT REGISTER SRO
96      177574      SR1=177574      ;" " " " SR1
97      177576      SR2=177576      ;" " " " SR2
98      172516      SR3=172516      ;ADDRESS OF MEM MGMT REGISTER SR3
99
100     172300      KIPDR0=172300      ;ADDRESS OF KERNEL 'I' PAGE
101     172302      KIPDR1=172302      ;DESCRIPTOR REGISTERS
102     172304      KIPDR2=172304
103     172306      KIPDR3=172306
104     172310      KIPDR4=172310
105     172312      KIPDR5=172312
106     172314      KIPDR6=172314
107     172316      KIPDR7=172316
108
109     172340      KIPAR0=172340      ;ADDRESSES OD KERNEL 'I' SPACE
110     172342      KIPAR1=172342      ;PAGE ADRESS REGISTERS
111     172344      KIPAR2=172344
112     172346      KIPAR3=172346

```

```

113          172350          KIPAR4=172350
114          172352          KIPAR5=172352
115          172354          KIPAR6=172354
116          172356          KIPAR7=172356
117
118
119          ;INSTRUCTION EQUATES
120          104400          HLT=TRAP
121          104C00          SCOPE=EMT          ;SCOPE IS AN EMT TRAP
122
123          ;MISC. EQUATES
124          000006          RW=6          ;R/W BIT IN PDR REGISTERS
125          000000          UP=0          ;UP BIT IN PDR REGISTERS
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142          000000          .=0
143          000000          .WORD 0          ;SPECIAL TRAP/INTERRUPT CATCHER IF PRO-
144          000002          .WORD 0          ;GRAM HALTS AT 0 THEN ADDRESS WAS NOT
145                                     ;LOADED PROPERLY FROM VECTOR.
146          000004          001126          .WORD ERRTRP
147          000006          000002          .WORD RTI
148          000034          000034          .=TRAPVEC
149          000034          001204          .WORD ERROR
150          000036          000340          .WORD PRTY7
151          000046          000046          .=46
152          000046          004260          $ENDAD
153
154          000052          000052          .=52
155          000052          040000          40000
156          000100          000100          .=100
157          000100          004567          000664          CRLF: JSR R5,$SPRINT
158          000104          000746          $CRLF
159          000106          000207          RTS PC
160          000110          000000          RELFL: .WORD 0
161          000112          000000          SAVPC2: .WORD 0
162          000162          000162          .=162
163          000162          012706          000500          PONE: MOV #500,SP ;STARTING ADDRESS TO RELOCATE LOADERS.
164          000166          004767          002016          JSR PC,$ALDR
165          000172          000000          HALT
166          000174          000401          BR PTW0
167          000200          000200          .=200
168          000200          012706          000500          PTW0: MOV #500,SP ;STARTING ADDRESS OF MEMORY TEST.

```

```

169 000204 000137 002376      JMP      @#START          ;GO TO START OF TEST
170                               ;=250
171 000250 000000      .WORD    0              ;MEMORY MANAGEMENT TRAP VECTOR.
172 000252 000000      .WORD    0
173                               ;
174                               ;ROUTINE TO SAVE REGISTERS ON THE STACK
175                               ;CALLED BY SAVE MACRO OR JSR   PC,$SAVR
176                               $SAVR: MOV      (SP)+,1$      ;SAVE RETURN PC
177 000254 012667 000016      MOV      R5,-(SP)
178 000260 010546      MOV      R4,-(SP)
179 000262 010446      MOV      R3,-(SP)
180 000264 010346      MOV      R2,-(SP)
181 000266 010246      MOV      R1,-(SP)
182 000270 010146      MOV      R0,-(SP)
183 000272 010046      MOV      (PC)+,PC      ;RETURN
184 000274 012707      1$:      0              ;CONTAINS RETURN ADDRESS
185 000276 000000
186                               ;ROUTINE TO RESTORE REGISTERS SAVED ON THE STACK
187                               ;CALLED BY RESTORE MACRO OR JSR PC,$RESTR
188                               $RESTR: MOV     (SP)+,1$      ;SAVE RETURN PC
189 000300 012667 000016      MOV     (SP)+,R0
190 000304 012600      MOV     (SP)+,R1
191 000306 012601      MOV     (SP)+,R2
192 000310 012602      MOV     (SP)+,R3
193 000312 012603      MOV     (SP)+,R4
194 000314 012604      MOV     (SP)+,R5
195 000316 012605      MOV     (PC)+,PC      ;RETURN
196 000320 012707      1$:      0              ;CONTAINS RETURN ADDRESS
197 000322 000000
198                               .SBTTL  POWER FAIL ROUTINE
199                               ;=502
200                               000502
201                               ;POWER FAIL ROUTINE
202                               ;THE POWER DOWN ROUTINE SAVES THE KEYBOARD STATUS,THE GENERAL REGISTERS
203                               ; (R0-R5) AND MEM MGMT REGISTERS (KIPDR0-KIPDR7,KIPAR0-KIPAR7,SR3,SR2,SR0)
204                               ;ON THE STACK AND SAVES THE STACK POINTER IN PFSTK BELOW.
205 000502 013746 177560      PDWN:  MOV     @#TKS,-(SP)      ;SAVE KEYBOARD STATUS
206 000506 004767 177542      JSR     PC,$SAVR          ;GO SAVE REGISTERS ON THE STACK
207 000512 005737 000762      TST     @#MMAVA         ;CHECK IF MEM MGMT IS AVAILABLE
208 000516 001421      BEQ     3$              ;BRANCH IF NOT AVAILABLE
209 000520 013746 177572      MOV     @#SR0,-(SP)      ;SAVE SR0
210 000524 013746 177576      MOV     @#SR2,-(SP)      ;SAVE SR2
211 000530 013746 172516      MOV     @#SR3,-(SP)      ;SAVE SR3
212 000534 012700 172300      MOV     @#KIPDR0,R0      ;GET ADDRESS OF KIPDR0
213 000540 012702 000010      MOV     #8,R2
214 000544 010203      MOV     R2,R3
215 000546 012046      1$:      MOV     (R0)+,-(SP)      ;SAVE KIPDR0-KIPDR7
216 000550 077202      SOB     R2,1$
217 000552 012700 172340      MOV     @#KIPAR0,R0      ;GET ADDRESS OF KIPAR0
218 000556 012046      2$:      MOV     (R0)+,-(SP)      ;SAVE KIPAR0-KIPAR7
219 000560 077302      SOB     R3,2$
220 000562 010627      3$:      MOV     SP,(PC)+        ;SAVE STACK PTR IN FOLLOWING LOCATION
221 000564 000000      PFSTK: .WORD    0        ;CONTAINS STACK PTR AFTER POWER FAIL
222 000566 012737 000576 000024      MOV     @#PUP,@#PFVEC    ;SET POWER FAIL VECTOR TO PUP ROUTINE
223 000574 000000
224

```

```

225      ;POWER UP ROUTINE.
226      PUP:  NOP
227      000576 000240      MOV      2#PFSTK,SP      ;SET STACK PTR
228      000600 013706 000564      TST      MMAVA          ;CHECK IF MEM MGMT IS AVAILABLE
229      000604 005767 000152      BEQ      4$
230      000610 001421      BEQ      4$
231      000612 012700 172360      MOV      #KIPAR7+2,R0    ;GET ADDRESS OF KIPAR7+2
232      000616 012702 000010      MOV      #8,R2
233      000622 010203      MOV      R2,R3
234      000624 012640      1$:  MOV      (SP)+,-(R0)      ;RESTORE KIPAR7-KIPAR0
235      000626 077302      SOB      R3,1$
236      000630 012700 172320      MOV      #KIPDR7+2,R0    ;GET ADDRESS OF KIPDR7+2
237      000634 012640      2$:  MOV      (SP)+,-(R0)      ;RESTORE KIPDR7-KIPDR0
238      000636 077202      SOB      R2,2$
239      000640 012637 172516      MOV      (SP)+,2#SR3      ;RESTORE SR3
240      000644 012637 177576      MOV      (SP)+,2#SR2      ;RESTORE SR2
241      000650 012637 177572      MOV      (SP)+,2#SRO      ;RESTORE SRO
242      000654 005767 004604      4$:  TST      PARAVA          ;CHECK IF PARITY REGISTERS ARE ENABLED
243      000660 001402      BEQ      5$              ;BRANCH IF NOT
244      000662 004767 004474      JSR      PC,.MAMF        ;GO ENABLE PARITY REGISTERS
245      000666 004767 177406      5$:  JSR      PC,$RESTR        ;RESTORE REGISTERS FROM STACK
246      000672 012637 177560      MOV      (SP)+,2#TKS
247      000676 012737 000502 000024      MOV      #PDWN,2#PFVEC    ;SET POWER FAIL TRAP TO PDWN ROUTINE
248      000704 005027      CLR      (PC)+
249      000706 000000      10$: .WORD 0
250      000710 005267 177772      11$: INC      10$            ;DELAY WAITING FOR TTY MOTOR
251      000714 100375      BPL      11$
252      000716 004567 000046      JSR      R5,$PRINT        ;GO TO PRINT ROUTINE
253      000722 000730      PWRFAIL
254      000724 000240      6$:  NOP
255      000726 000002      RTI              ;RETURN
256
257      000730 005015 047520 042527 PWRFAIL:.ASCII <15><12>'POWER FAILED'
258      000736 020122 040506 046111
259      000744 042105
260      000746 005015 000      $CRLF: .ASCIZ <15><12>
261
262
263      .SBTTL TAGS & PRINT ROUTINE
264
265      000752 000000      ICNT:  .WORD 0          ;CONTAINS PASS COUNT
266      000754 000000      ICOUNT: .WORD 0        ;CONTAINS ITERATION PATTERN
267      000756 000000      ERCNT:  0              ;CONTAINS ERROR COUNT
268      000760 000000      LDDISP: 0              ;CONTAINS DISPLAY REGISTER IMAGE
269      000762 000000      MMAVA:  0              ;MEM MGMT AVAILABLE INDICATOR
270
271
272
273      000764 000000      RELOCF: .WORD 0        ;CONTAINS RELOCATION FACTOR
274      000766 000000      COUNT:  .WORD 0        ;TEMPORARY WORKING LOCATION
275
276      ;ROUTINE TO PASS MESSAGE ADDRESS TO TYPE ROUTINE BELCW
277      ;CALL: JSR      R5,$PRINT
278      ;      MESSAGE ADDRESS
279      000770 000240      $PRINT: NOP
280      000772 012567 000016      MOV      (R5)+,1$      ;GET MESSAGE ADDRESS
    
```

```

281 000776 066767 177762 000010      ADD      RELOCF,1$      ;ADD RELOCATION FACTOR
282 001004 013746 177776              MOV      2#PSW,-(SP)   ;PUSH PSW ON THE STACK
283 001010 004767 000014              JSR      PC,.TYPE     ;CALL TYPE ROUTINE
284 001014 000000      1$:      WORD      0      ;CONTAINS MESSAGE ADDRESS
285 001016 000205      RTS      R5            ;RETURN
286
287 ;ROUTINE TO TYPE ASCII MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
288 ;THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
289 ;CALL:  TYPE
290 ;
291 ;           MESADR           ;MESADR IS FIRST ADDRESS OF ASCIZ STRING
292
293 ;TAGS USED BY THE TYPE ROUTINE BELOW
294 $NULL: .BYTE 0          ;CONTAINS NULL CHARACTER
295 $FILL: .BYTE 2          ;CONTAINS # OF FILLER CHARACTERS
296 $TPFLG: .BYTE 0         ;CONTAINS TELEPRINTER AVAILABLE FLAG
297 ;0/377 = AVAIL/NOT AVAIL
298 $TKFLG: .BYTE 0         ;CONTAINS KEYBOARD AVAILABLE FLAG
299 $TPS:   .WORD 177564     ;ADDRESS OF TELEPRINTER STATUS REGISTER
300 $TPB:   .WORD 177566     ;ADDRESS OF TELEPRINTER DATA BUFFER
301 .TYPE:  MOV      R0,-(SP) ;SAVE R0
302         MOV      2(SP),R0 ;GET MESSAGE ADDRESS
303         ADD      #2,2(SP)  ;ADJUST RETURN PC
304 1$:     MOV      (R0)+,-(SP) ;PUSH CHARACTER TO BE TYPED ONTO STACK
305         BNE     2$         ;BRANCH IF NOT THE TERMINATOR
306         TST     (SP)+      ;POP TERMINATOR CHAR OFF THE STACK
307         MOV      (SP)+,R0  ;RESTORE R0
308         RTI                    ;RETURN TO CALLER
309
310 2$:     JSR      PC,TYPIT   ;TYPE CHARACTER
311 3$:     CMP      #12,(SP)+  ;CHECK IF CHARACTER WAS A LINE FEED
312         BNE     1$         ;BRANCH IF NOT LINE FEED
313         MOV      $NULL,-(SP) ;GET # OF FILLERS REQUIRED AND FILLER
314 ;CHARACTER.
315
316 4$:     DECB     1(SP)      ;DECREMENT FILLERS REQ. COUNT
317         BLT     3$         ;BRANCH IF NO MORE FILLERS ARE REQUIRED
318         JSR      PC,TYPIT   ;TYPE FILLER CHARACTER
319         BR      4$
320
321 TYPIT:  TST      2$TPS     ;WAIT FOR OUTPUT DEVICE
322         BPL     .-4
323         MOV      2(SP),2$TPB ;OUTPUT CHARACTER
324         RTS      PC
325
326 ;XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
327 ;ERROR TRAP SERVICE ROUTINE
328 ERTRTP: TST      2#SWR     ;CHECK IF HALT ON ERROR
329         BPL     .+4        ;BRANCH IF NO HALT ON ERROR
330         HALT
331         TST     (PC)+      ;CHECK IF PREV TRAP TO 4 REPORTED
332 1$:     .WORD   0          ;CONTAINS ERROR REPORTED FLAG
333         BNE     2$         ;BRANCH IF NOT REPORTED
334         MOV     SP,1$      ;SET 'NOT REPORTED'
335         MOV     (SP),R2    ;GET PC OFF STACK
336         JSR     PC,$FORMO  ;GO TO FORMAT ROUTINE

```

```

337 001156 004567 177606 JSR R5,$SPRINT ;GO TO PRINT ROUTINE
338 001162 001460 TRAP4
339 001164 004567 177600 JSR R5,$SPRINT ;GO TO PRINT ROUTINE
340 001170 002351 DIGITS
341 001172 000000 2$: HALT ;ERROR! SECOND TRAP TO 4 OCCURRED
342 ;BEFORE FIRST WAS PRINTED
343 001174 005067 177740 CLR 1$
344 001200 000137 000200 JMP @#200 ;RESTART AT 200
345
346 ;SBTTL ERROR SERVICE ROUTINE
347 ;ERROR SERVICE CALLED BY JSR PC,ERROR INSTRUCTION
348 ;OR HLT (A TRAP INST)
349 001204 000240 ERROR: NOP
350 001206 022767 017777 177542 CMP #17777,ERCNT ;CHECK FOR MAX ERROR CNT
351 001214 001403 BEQ 4$
352 001216 062767 000001 177532 ADD #1,ERCNT ;INCREMENT ERROR COUNT
353 001224 032737 001000 177570 4$: BIT #BIT9,@#SWR ;SWITCH 9 UP?
354 001232 001411 BEQ 5$
355 001234 042767 017777 177516 BIC #17777,LODISP ;SAVE RELOCATION BITS
356 001242 056767 177510 177510 BIS ERCNT,LODISP ;LOAD ERROR COUNT
357 001250 016737 177504 177570 MOV LODISP,@#DISPLAY ;LOAD DISPLAY REGISTER
358 001256 005737 177570 5$: TST @#SWR ;HALT ON ERROR
359 001262 100002 BPL .+6
360 001264 000000 HALT
361 001266 000470 BR 3$
362 001270 032737 020000 177570 BIT #20000,@#SWR ;PRINT OUT DESIRED?
363 001276 001051 BNE 1$ ;BRANCH IF NO PRINTOUT
364 001300 004767 176750 JSR PC,$SAVR ;GO SAVE REGISTERS ON THE STACK
365 001304 016602 000014 MOV 14(SP),R2 ;GET PC OF ERROR CALL
366 001310 004767 000240 JSR PC,$FORMO ;GO TO FORMAT ROUTINE
367 001314 004567 177450 JSR R5,$SPRINT ;GO TO PRINT ROUTINE
368 001320 001475 ERRPC
369 001322 004567 177442 JSR R5,$SPRINT ;GO TO PRINT ROUTINE
370 001326 002351 DIGITS
371 001330 016602 000004 MOV 4(SP),R2 ;GET FAILING ADDRESS (IN R2)
372 001334 004767 000214 JSR PC,$FORMO ;GO TO FORMAT ROUTINE
373 001340 004567 177424 JSR R5,$SPRINT ;GO TO PRINT ROUTINE
374 001344 002327 ADDRESS
375 001346 105767 003175 TSTB PENFLG ;BRANCH IF PARITY ERROR DETECTED
376 001352 001017 BNE 11$ ;BUT NOT FOUND
377 001354 105767 003166 TSTB PEFLG ;BRANCH IF PARITY ERROR DETECTED
378 001360 001006 BNE 10$ ;BUT FOUND
379 001362 004567 177402 JSR R5,$SPRINT ;GO TO PRINT ROUTINE
380 001366 001501 XMTDAT
381 001370 010046 MOV R0,-(SP) ;PUSH VALUE TO TYPED ONTO STACK
382 001372 004767 000416 JSR PC,02A ;GO PRINT VALUE
383 001376 10$: JSR R5,$SPRINT ;GO TO PRINT ROUTINE
384 001376 004567 177366 JSR R5,$SPRINT ;GO TO PRINT ROUTINE
385 001402 001514 RECDAT
386 001404 010346 MOV R3,-(SP) ;PUSH VALUE TO BE TYPED ONTO STACK
387 001406 004767 000402 JSR PC,02A
388 001412 004767 176462 11$: JSR PC,CRLF
389 001416 004767 176656 JSR PC,$RESTR ;RESTORE REGISTERS FROM STACK
390 001422 032737 002000 177570 1$: BIT #2000,@#SWR ;RING BELL ON ERROR
391 001430 001403 BEQ 2$
392 001432 004567 177332 JSR R5,$SPRINT ;GO TO PRINT ROUTINE
    
```

393	001436	001527			BELL		
394	001440	005737	177570		2\$: TST	@#SWR	;HALT AFTER PRINT OUT
395	001444	100001			BPL	.+4	
396	001446	000000			HALT		
397	001450	010042			3\$: MOV	R0,-(R2)	;RESTORE CORRECT DATA TO ADDRESS
398	001452	062702	000002		ADD	#2,R2	
399	001456	000002			RTI		
400							
401	001460	051124	050101	042520	TRAP4:	.ASCII	'TRAPPED TO 4'
402	001466	020104	047524	032040			
403	001474	040					
404	001475	120	036503	000	ERRPC:	.ASCIZ	'PC='
405	001501	107	047517	020104	XMTDAT:	.ASCIZ	'GOOD DATA='
406	001506	040504	040524	000075			
407	001514	041040	042101	042040	RECDAT:	.ASCIZ	'BAD DATA='
408	001522	052101	036501	000			
409	001527	007	000		BELL:	.ASCIZ	<?>
410	001531	120	051101	052111	PARREG:	.ASCIZ	/PARITY ERROR REG=/'
411	001536	020131	051105	047522			
412	001544	020122	042522	036507			
413	001552	000					
414		001554					
415					.EVEN		
416	001554	066767	177204	000014	:ROUTINE	TO PLACE ASCII VALUE OF AN ADDRESS IN TO ADDRESS MESSAGE	
417	001562	066767	177176	000152	\$FORM0:	ADD	RELOC,11\$+2
418	001570	004767	176460		ADD	RELOC,41\$+2	
419	001574	012704	002351		JSR	PC,\$\$AVR	;GO SAVE REGISTERS ON THE STACK
420	001600	005003			11\$: MOV	#DIGITS,R4	;ADDRESS WHERE ASCII VALUES ARE STORED
421	001602	162702	000002		CLR	R3	;WORKING & INDEX REGISTER
422	001606	010205			SUB	#2,R2	;ADJUST ADDRESS
423	001610	010501			MOV	R2,R5	;SAVE
424	001612	005767	177144		MOV	R5,R1	
425	001616	001426			TST	MMAVA	;CHECK IF MEM MGMT IS AVAILABLE
426	001620	032737	000001	177572	BEQ	\$;BRANCH IF NOT AVAILABLE
427	001626	001422			BIT	#1,@#SRO	;IS MEM MGMT ENABLED
428	001630	042701	017777		BEQ	1\$	
429	001634	000301			BIC	#17777,R1	;SAVE PAR SELECTOR BITS
430	001636	006001			SWAB	R1	;SWAP BYTES
431	001640	006001			ROR	R1	
432	001642	006001			ROR	R1	;FORM INDEX VALUE
433	001644	006001			ROR	R1	
434	001646	017102	001774		MOV	@PARTAB(1),R2	;GET CONTENTS OF PAR
435	001652	012700	000006		MOV	#6,R0	;SHIFT COUNT
436	001656	006302			ASL	R2	;SHIFT KIPARI 6 PLACES LEFT
437	001660	006103			ROL	R3	;MSB'S GO INTO R3
438	001662	077003			SQB	R0,-4	
439	001664	042705	160000		BIC	#160000,R5	;CLEAR PAR SELECTOR BITS
440	001670	060502			ADD	R5,R2	;FORM 22 BIT ADDRESS
441	001672	005503			ADC	R3	;IN R2 & R3
442	001674	005001			1\$: CLR	R1	
443	001676	012700	000005		MOV	#5,R0	
444	001702	006003			12\$: ROR	R3	
445	001704	006002			ROR	R2	
446	001706	006001			ROR	R1	
447	001710	005300			DEC	R0	
448	001712	001373			BNE	12\$	


```

449 001714 012700 000010          MOV    #8,R0          ;DIGIT COUNT
450 001720 000405                BR     3$             ;PRINT FIRST DIGIT
451 001722 006301          2$:  ASL    R1
452 001724 006102          ROL    R2
453 001726 006103          ROL    R3
454 001730 005305          DEC    R5
455 001732 001373          BNE    2$
456 001734 012705 000003          MOV    #3,R5          ;DIGIT SHIFT COUNT
457 001740 116324 002312          41$: MOVB  DIGTAB(3),(4)+ ;LOAD DIGIT INTO MESSAGE
458 001744 005003          CLR    R3             ;CLEAR INDEX
459 001746 005300          DEC    R0             ;DEC DIGIT COUNT
460 001750 001364          BNE    2$
461 001752 004767 176322          JSR    PC,$RESTR      ;RESTORE REGISTERS FROM STACK
462 001756 046767 177002 177612          BIC    RELOC,11$+2
463 001764 046767 176774 177750          BIC    RELOC,41$+2
464 001772 000207          RTS     PC             ;RETURN
465
466 001774 172340          PARTAB: KIPAR0
467 001776 172342          KIPAR1
468 002000 172344          KIPAR2
469 002002 172346          KIPAR3
470 002004 172350          KIPAR4
471 002006 172352          KIPAR5
472 002010 172354          KIPAR6
473 002012 172356          KIPAR7
474
475          ;ROUTINE TO TYPE OCTAL VALUE PUSHED ONTO STACK
476          ;CALL: MOV    VALUE, -(SP)      ;PUSH VALUE ONTO STACK
477          ;      JSR    PC,02A           ;CALL ROUTINE
478
479 002014          02A: JSR    PC,$SAVR      ;GO SAVE REGISTERS ON THE STACK
480 002020 004767 176234          MOV    16(SP),R0     ;GET VALUE
481 002024 012703 000016          MOV    #6,R3         ;COUNTER
482 002030 005002          CLR    R2             ;WORKING REGISTER
483 002032 006100          ROL    R0
484 002034 006102          ROL    R2
485 002036 062702 000260          1$:  ADD    #260,R2     ;FORM ASCII VALUE
486 002042 010267 000040          MOV    R2,2$         ;MOVE CHAR TO TYPE LOCATION
487 002046 004567 176716          JSR    R5,$PRINT     ;GO TO PRINT ROUTINE
488 002052 002106          2$:  CLR    R2
489 002054 005002          ROL    R0
490 002056 006100          ROL    R2
491 002060 006102          ROL    R0
492 002062 006100          ROL    R2
493 002064 006102          ROL    R0
494 002066 006100          ROL    R2
495 002070 006102          ROL    R0
496 002072 005303          DEC    R3
497 002074 001360          BNE    1$
498 002076 004767 176176          JSR    PC,$RESTR      ;RESTORE REGISTERS FROM STACK
499 002102 012616          MOV    (SP)+,(SP)
500 002104 000207          RTS     PC
501 002106 000000          2$:  .WORD 0           ;CONTAINS CHARACTER TO BE TYPED
502
503 002110 000000          LODFLU: .WORD 0
504          ;ROUTINE TO SAVE ABS LOADER
    
```

```

505 002112 005767 177772 $LDR: TST LODFLO
506 002116 001401 BEQ 3$
507 002120 000207 RTS PC
508 002122 012700 017776 3$: MOV #17776,R0
509 002126 012737 002140 000004 MOV #2$,2#ERRVEC ;SET TIME OUT TRAP VECTOR
510 002134 005720 TST (R0)+
511 002136 000776 BR -2
512 002140 022626 2$: CMP (SP)+,(SP)+
513 002142 022700 020000 CMP #20000,R0 ;4K MACHINE?
514 002146 001417 BEQ 4$ ;YES--GET OUT
515 002150 162700 005672 SUB #1500,+1*2,R0 ;POINT R0 BACK TO LOADER
516 002154 010067 000102 MOV R0,$LDR1 ;SAVE FOR RESTORE ROUTINE
517 002160 012702 002734 MOV #1500,R2 ;WORD COUNT
518 002164 012703 010200 MOV #LODAR,R3 ;WHERE LOADER IS TO BE STORED
519 002170 012023 1$: MOV (R0)+,(R3)+ ;STORE LOADER
520 002172 005302 DEC R2
521 002174 001375 BNE 1$
522 002176 014367 000042 MOV -(R3),LSTLOC ;SAVE LAST WORD OF LOADERS
523 002202 005367 177702 DEC LODFLO
524 002206 000207 4$: RTS PC ;RETURN
525
526 ;ROUTINE TO RESTORE LOADER
527 002210 005767 177674 $RLDR: TST LODFLO
528 002214 001001 BNE 2$
529 002216 000207 RTS PC
530 002220 016705 000036 2$: MOV $LDR1,R5 ;GET FIRST ADDRESS OF WHERE LOADER IS
531 ;TO BE RESTORED
532 002224 012704 010200 MOV #LODAR,R4 ;ADDRESS WHERE LOADER IS STORED
533 002230 012702 002734 MOV #1500,R2 ;WORD COUNT
534 002234 012425 1$: MOV (R4)+,(R5)+ ;RESTORE
535 002236 005302 DEC R2
536 002240 001375 BNE 1$
537 002242 012745 MOV (PC)+,-(R5) ;RESTORE LAST LOCATION (SAVED BY SAVE
538 002244 000000 LSTLOC: .WORD 0 ;LOADERS ROUTINE ABOVE)
539 002246 004567 176516 JSR RS,$PRINT ;GO TO PRINT ROUTINE
540 002252 002264 $LDRM
541 002254 005067 177630 CLR LODFLO
542 002260 000207 RTS PC ;RETURN TO CALLER
543
544 002262 000000 $LDR1: .WORD 0 ;FIRST ADDRESS WHERE LOADERS ARE TO BE
545 ;RESTORED TO
546 002264 047514 042101 051105 $LDRM: .ASCIZ 'LOADER IS RESTORED'<15><12>
547 002272 044440 020123 042522
548 002300 052123 051117 042105
549 002306 005015 000
550 002312 .EVEN
551 ;DIGIT TABLE
552 002312 030460 DIGTAB: "01
553 002314 031462 "23
554 002316 032464 "45
555 002320 033466 "67
556
557 ;MESSAGES
558 002322 040514 052123 040 LST: .ASCII 'LAST '
559 002327 115 046505 051117 ADDRESS: .ASCII 'MEMORY ADDRESS IS '
560 002334 020131 042101 051104
    
```

561 002342 051505 020123 051511
 562 002350 040
 563 002351 060 030060 030060
 564 002356 030060 060
 565 002361 040 000
 566 002363 120 051501 036523
 567 002370 020040 000
 568 002374 002374
 569 002374 000000
 570
 571
 572
 573
 574
 575
 576
 577
 578
 579
 580
 581
 582
 583
 584 002376 012737 002440 000212
 585 002404 012706 000500
 586 002410 004767 177476
 587 002414 004567 176350
 588 002420 007520
 589 002422 005037 000756
 590 002426 005037 000760
 591 002432 013737 000760 177570
 592 002440 012706 000500
 593 002444 005037 004546
 594 002450 052737 000014 177746
 595 002456 012727 002440
 596 002462 000000
 597 002464 005037 000752
 598 002470 005037 000764
 599 002474 012737 000502 000024
 600 002502 005037 000026
 601
 602
 603 002506 005067 176250
 604 002512 032737 010000 177570
 605 002520 001020
 606 002522 012737 002562 000004
 607 002530 005037 177572
 608 002534 005167 176222
 609 002540 012737 000020 172516
 610 002546 022737 000020 172516
 611 002554 001002
 612 002556 006367 176200
 613 022562 004767 002574
 614
 615
 616

DIGITS: .ASCII '00000000'
 SPACE1: .ASCIZ ' '
 PASSMG: .ASCII 'PASS='
 PASSNM: .ASCIZ ' '
 .EVEN
 PLACE: .WORD 0
 .SBTTL MEMORY ADDRESS TESTS

; THIS TEST ADDRESS MEMORY UP TO 128K AND PROVES 'UNIQUENESS' OF ALL
 ; MEMORY ADDRESS IN A 32K SEGMENT. THE TEST WRITES INTO EACH MEMORY
 ; ADDRESS THE VALUE OF THAT ADDRESS AND THEN CHECKS FOR THE CORRECT
 ; DATA IN EACH ADDRESS.
 ; THE TWELVE MOST SIGNIFICANT BITS OF THE LAST AVAILABLE MEMORY ADDRESS
 ; IS STORED IN RS.
 ; STARTING INSTRUCTIONS
 ; LOAD ADDRESS=200
 ; PRESS START
 ; STACK POINTER IS AT 500
 ; *****RESTART AT 162 TO RESTORE LOADER*****

MEMORY ADDRESS TEST
 START: MOV #START1, #212 ; CHANGE START ADDRESS
 MOV #STKPTR, SP ; SET UP STACK PTR
 JSR PC, \$LDR ; GO SAVE MONITOR & LOADERS
 JSR RS, \$PRINT ; GO TO PRINT ROUTINE
 RESLDR
 CLR #ERCNT ; CLEAR ERROR COUNT
 CLR #LDDISP ; CLEAR DISPLAY REGISTER STORAGE LOCN
 MOV #LDDISP, #DISPLAY ; CLEAR DISPLAY REGISTER
 START1: MOV #STKPTR, SP ; SET STACK PTR
 CLR #PEFLG ; CLEAR PARITY ERROR INDICATORS
 BIS #14, #CNTRL ; DISABLE CACHE
 MOV #START1, (PC)+ ; LOAD PARITY ERROR RESTART ADDRESS
 PERSTR: .WORD 0 ; CONTAINS RESTART ADDRESS AFTER PAR ERR
 CLF #ICNT ; CLEAR PASS COUNT
 CLR #RELOCF ; CLEAR RELOCATION FACTOR
 MOV #PDWN, #PFVEC ; SET POWER FAIL TRAP VECTOR
 CLR #PFVEC+2

; CHECK IF MEMORY MANAGEMENT IS AVAILABLE
 CLR MAVA ; CLEAR MEM MGMT AVAILABLE INDICATOR
 BIT #BIT12, #SWR ; CHECK IF TO RUN WITH MEM MGMT
 BNE 1\$; DO NOT USE MEM MGMT IF SW12 WAS SET
 MOV #15, #ERRVEC ; SET TIME OUT TRAP
 CLR #SR0 ; REFERENCE MEM MGMT
 COM MAVA ; SET INDICATOR TO -1 IF AVAILABLE
 MOV #20, #SR3 ; SET 22 BIT MODE
 CMP #20, #SR3 ; DID IT SET?
 BNE 1\$; NO--BRANCH
 ASL MAVA ; YES--SET INDICATOR TO -2
 1\$: JSR PC, .MAMF ; GO ENABLE PARITY ACTION

; ROUTINE TO WRITE VALUE OF MEMORY ADDRESS INTO MEMORY ADDRESS

```

617 ;FOR EXAMPLE ROUTINE WRITES 20000 INTO LOCATION 20000
618 002566 012737 002626 000004 WRTUP: MOV #DONE0,2#ERRVEC ;SET TIME OUT TRAP VECTOR
619 002574 010701 MOV PC,R1 ;LOAD TRACE REGISTER
620 002576 004767 002712 JSR PC,LDMMO
621 002602 012737 005616 000250 MOV #MMABTO,2#MMVEC ;SET MEM MGMT ABORT VECTOR
622 002610 012702 020000 MOV #20000,R2 ;FIRST ADDRESS
623 002614 010203 MOV R2,R3 ;LOAD CONSTANT
624 002616 010322 1$: MOV R3,(R2)+ ;WRITE VALUE OF ADDRESS INTO ADDRESS
625 002620 062703 000002 ADD #2,R3 ;NEXT VALUE
626 002624 000774 BR 1$ ;WRITE UNTIL DONE
627
628 002626 012706 000500 DONE0: MOV #STKPTR,SP ;SET STACK PTR
629 002632 004767 176716 JSR PC,$FORM0 ;GO TO FORMAT ROUTINE
630 002636 004567 176126 JSR RS,$PRINT ;GO TO PRINT ROUTINE
631 002642 002322 LST
632 002644 004767 175230 JSR PC,CRLF
633
634 ;ROUTINE TO CHECK THAT VALUE OF MEMORY ADDRESS WAS WRITTEN CORRECTLY
635 002650 010701 MOV PC,R1 ;LOAD TRACE REGISTER
636 002652 012702 020000 MOV #20000,R2 ;SET R2
637 002656 012737 002732 000004 MOV #DONE1,2#ERRVEC ;SET TIME OUT TRAP
638 002664 010200 MOV R2,R0
639 002666 162700 000002 SUB #2,R0 ;SUBTRACT 2
640 002672 004767 002616 JSP PC,LDMMO
641 002676 062700 000002 1$: ADD #2,R0
642 002702 012203 MOV (R2)+,R3 ;GET WRITTEN VALUE
643 002704 020003 CMP R0,R3 ;CHECK
644 002706 001402 BEQ 2$
645 002710 104400 HLT ;ERROR! TO DETERMINE WHICH ADDRESS WAS
646 002712 000771 BR 1$
647 002714 005142 2$: COM -(R2)
648 002716 005112 COM (R2)
649 002720 012203 MOV (R2)+,R3
650 002722 020003 CMP R0,R3
651 002724 001764 BEQ 1$
652 002726 104400 HLT
653 ;WRITTEN IMPROPERLY EXAMINE R2. NEXT EXAMINE MEM MGMT REGISTER KIPARI
654 ;(IF MEM MGMT IS AVAILABLE). ADD R2 AND KIPARI TOGETHER AS SHOWN BELOW
655
656 : R2-2 0 00X XXX XXX XXX XXX
657 : KIPARI(772342) Y YYY YYY YYY YYY YYY
658 : ADDRESS Z ZZZ ZZZ ZZZ ZZZ ZZZ ZZZ
659
660 002730 000762 BR 1$
661 002732 012706 000500 DONE1: MOV #STKPTR,SP ;SET STACK PTR
662 002736 010701 MOV PC,R1 ;LOAD TRACE REGISTER
663
664 ;ROUTINE TO WRITE 1'S COMPLEMENT VALUE OF ADDRESS INTO ADDRESS
665 ;FOR EXAMPLE ROUTINE WRITES 157777 INTO ADDRESS 20000
666
667 002740 005767 176016 TST MMAVA ;MEMORY MAGNAGEMENT AVAILABLE?
668 002744 001420 BEQ 3$
669 002746 013703 172342 MOV #KIPARI,R3 ;FIND LAST ADDRESS IF MEM MANAGE USED
670 002752 006303 ASL R3
671 002754 006303 ASL R3
672 002756 006303 ASL R3
    
```

```

673 002760 006303 ASL R3
674 002762 006303 ASL R3
675 002764 006303 ASL R3
676 002766 010246 MOV R2, -(SP) ; DEVELOP COMPLEMENT OF LAST ADDRESS
677 002770 042716 020000 BIC #20000, (SP) ; SAVE BITS IF MEMORY IS NOT A MULTIPLE OF 4K
678 002774 062603 ADD (SP)+, R3
679 002776 012737 005650 000250 MOV #MMABT1, #MMVEC ; SET ABORT VECTOR
680 003004 000403 BR 2$
681 003006 162702 000002 3$: SUB #2, R2 ; R2=LAST ADDRESS
682 003012 010203 MOV R2, R3
683 003014 005103 2$: COM R3 ; COMPLEMENT VALUE IN R3
684 003016 062703 000002 1$: ADD #2, R3
685 003022 010342 MOV R3, -(R2) ; WRITE COMPLIMENT VALUE INTO ADDRESS
686 003024 102403 BVS DONE3
687 003026 020227 017776 CMP R2, #17776
688 003032 001371 BNE 1$
689
690 ; SET UP TO CHECK COMPLEMENT DATA WRITTEN DOWN
691 003034 000240 DONE3: NOP
692 003036 010701 MOV PC, R1 ; LOAD TRACE REGISTER
693 003040 005767 175716 TST MMABA ; CHECK IF MM IS AVAIL
694 003044 001406 BEQ 1$
695 003046 012737 000200 172342 MOV #200, #KIPARI ; INIT KIPARI
696 003054 012737 005616 000250 MOV #MMABT0, #MMVEC ; SET ABORT VECTOR
697 003062 012737 003122 000004 1$: MOV #DONE4, #ERRVEC
698 003070 012702 020000 MOV #20000, R2 ; FIRST ADDRESS
699 003074 010200 MOV R2, R0
700 003076 005100 COM R0 ; FIRST DATA (COM OF ADDRESS)
701 003100 062700 000002 ADD #2, R0
702 003104 162700 000002 2$: SUB #2, R0
703 003110 012203 MOV (R2)+, R3 ; GET VALUE
704 003112 020003 CMP R0, R3 ; CHECK
705 003114 001773 BEQ 2$
706 003116 104400 HLT
707 003120 000771 BR 2$
708 003122 000240 DONE4: NOP
709
710 ; ROUTINE TO WRITE BANK # INTO ALL ADDRESSES IN A 4K BANK
711 003124 012737 003172 000004 MOV #DONE4A, #ERRVEC ; SET TIME OUT TRAP VECTOR
712 003132 010701 MOV PC, R1
713 003134 004767 002354 JSR PC, LDMMO
714 003140 012737 005616 000250 MOV #MMABT0, #MMVEC
715 003146 012702 020000 MOV #20000, R2
716 003152 005000 CLR R0
717 003154 005200 1$: INC R0 ; R0 WILL BE DATA WRITTEN
718 003156 012704 010000 MOV #4096, R4 ; SET 4K COUNTER
719 003162 010022 2$: MOV R0, (R2)+ ; WRITE BANK # INTO ALL ADDRESSES
720 003164 005304 DEC R4
721 003166 001375 BNE 2$
722 003170 000771 BR 1$
723
724 003172 022626 DONE4A: CMP (SP)+, (SP)+ ; ADJUST STACK PTR
725
726 ; CHECK THAT DATA WRITTEN ABOVE CAN BE READ
727 003174 012737 003242 000004 MOV #DONE4B, #ERRVEC
728 003202 010701 MOV PC, R1
    
```

729	003204	004767	002304			JSR	PC, LDMMO	
730	003210	012702	020000			MOV	#20000, R2	
731	003214	005000				CLR	R0	
732	003216	005200			1\$:	INC	R0	
733	003220	012704	010000			MOV	#4096., R4	
734	003224	012203			2\$:	MOV	(R2)+, R3	
735	003226	020003				CMP	R0, R3	
736	003230	001401				BEQ	.+4	
737	003232	104400				HLT		
738	003234	005304				DEC	R4	
739	003236	001372				BNE	2\$	
740	003240	000766				BR	1\$	
741	003242	022626			DONE4B:	CMP	(SP)+, (SP)+	
742								
743								
744								
745	003244	010701				MOV	PC, R1	
746	003246	012737	005650	000250		MOV	#MMABT1, #MMVEC	
747	003254	162702	000002			SUB	#2, R2	
748	003260	005000				CLR	R0	
749	003262	005300			1\$:	DEC	R0	
750	003264	012704	010000			MOV	#4096., R4	
751	003270	010042			2\$:	MOV	R0, -(R2)	
752	003272	102406				BVS	DONE4C	
753	003274	020227	017776			CMP	R2, #17776	;CHECK IF DONE
754	003300	001403				BEQ	DONE4C	
755	003302	005304				DEC	R4	
756	003304	001371				BNE	2\$	
757	003306	000765				BR	1\$	
758								
759	003310	012737	003422	000004	DONE4C:	MOV	#DONE4D, #ERRVEC	
760	003316	010701				MOV	PC, R1	
761	003320	004767	002170			JSR	PC, LDMMO	
762	003324	012737	005616	000250		MOV	#MMABT0, #MMVEC	;SET ABORT VECTOR
763	003332	012702	020003			MOV	#20000, R2	
764	003336	022704	010000		1\$:	CMP	#4096., R4	;CHECK IF WRITE ABOVE STARTED ON ;4K BOUNDARY
765								
766	003342	001415				BEQ	2\$	
767	003344	012203				MOV	(R2)+, R3	
768	003346	020003				CMP	R0, R3	
769	003350	001402				BEQ	4\$	
770	003352	104400				HLT		
771	003354	000406				BR	5\$	
772	003356	005142			4\$:	COM	-(R2)	
773	003360	005112				COM	(R2)	
774	003362	012203				MOV	(R2)+, R3	
775	003364	020003				CMP	R0, R3	
776	003366	001401				BEQ	5\$	
777	003370	104400				HLT		
778	003372	005204			5\$:	INC	R4	
779	003374	001360				BNE	1\$	
780	003376	005200			2\$:	INC	R0	
781	003400	012704	010000			MOV	#4096., R4	
782	003404	012203			3\$:	MOV	(R2)+, R3	
783	003406	020003				CMP	R0, R3	
784	003410	001401				BEQ	.+4	

```

785 003412 104400          HLT
786 003414 005304          DEC      R4
787 003416 001372          BNE     3$
788 003420 000766          BR      2$
789
790 003422 022626          DONE4D: CMP      (SP)+,(SP)+
791 003424 005737 000042      TST     2#42          ;BRANCH IF PROGRAM WAS NOT
792 003430 001406          BEQ     BEGIN1        ;LOADED VIA ACT11 IN QV OR AA MODES
793 003432 005767 000624      TST     SENDAD+2      ;BRANCH IF NOT IN QV MODE
794 003436 100003          BPL     BEGIN1
795 003440 012737 000001 004212  MOV     21,2#ENDCT    ;SET ENDCT TO DO 1 PASS ONLY IN QV
796          .SBTTL  WORST CASE NOISE TESTS
797          ;THIS TEST WRITES MEMORY WORST CASE NOISE TEST PATTERNS THROUGHOUT
798          ;MEMORY AND CHECKS THAT THEY CAN BE WRITTEN AND READ.
799          ;SET UP TRAP VECTORS
800 003446 012706 000500      BEGIN1: MOV     #STKPTR,SP          ;SET STACK PTR
801 003452 052737 000014 177746  BIS     #14,2#CNTRL        ;DISABLE CACHE
802 003460 004767 001676          JSR     PC,.MAMF          ;GO ENABLE PARITY ACTION
803 003464 004767 003726          JSR     PC,CKSWR         ;GO CHECK SWITCHES
804 003470 005027          CLR     (PC)+           ;SET INDICATOR TO WRITE NORMAL 3X9 PAT
805 003472 000000          PARPAT: .WORD 0
806 003474 022767 177776 175260  CMP     #-2,MMAVA        ;22 BIT MODE
807 003502 001002          BNE     DONE6           ;NO--BRANCH
808 003504 004767 001756          JSR     PC,MARGIN        ;YES--GO SETUP MARGINS
809
810
811          ;WRITE 3 XOR 9 TEST PATTERN STARTING AT ADDRESS 20000
812          ;NOTE PATTERN IS NORMAL 3 XOR 9 IF NO PARITY MEMORY IS AVAILABLE,
813          ;AND IS A MODIFIED PATTERN IF PARITY MEMORY IS AVAILABLE.
814          ;THE CONTENTS OF PARPAT IF 0/NOT 0 INDICATE IF NORMAL/MODIFIED PATTERN
815          ;IS BEING USED IN TESTS BELOW.
816 003510 012706 000500      DONE6:  MOV     #STKPTR,SP          ;SET STACK PTR
817 003514 010701          MOV     PC,R1           ;UPDATE TRACE REGISTER
818 003516 012737 003536 000004  MOV     #DONE7,2#ERRVEC    ;SET TIME OUT TRAP VECTOR
819 003524 012746 000001          MOV     #1,-(SP)        ;PUSH STARTING BANK # ON STACK
820 003530 005046          CLR     -(SP)          ;PUSH # OF 256. WORD BLOCKS TO WRITE
821 003532 004767 002326          JSR     PC,.3X9         ;CALL ROUTINE TO WRITE 3XOR9 PATTERN
822
823          ;CHECK 3 XOR 9 TEST PATTERN WRITTEN ABOVE
824 003536 012737 001126 000004  DONE7:  MOV     #ERRTRP,2#ERRVEC
825 003544 016600 000006          MOV     6(SP),R0        ;GET # OF 256. WORD BLOCKS WRITTEN
826 003550 005400          NEG     R0              ;FORM TWO'S COMPLEMENT
827 003552 010027          MOV     R0,(PC)+        ;SAVE # OF 256 WORD BLOCKS
828 003554 000000          WDS.256: .WORD 0        ;CONTAINS # OF 256 WORD BLOCKS IN MEM.
829 003556 012706 000500      MOV     #STKPTR,SP          ;SET STACK PTR
830 003562 010701          MOV     PC,R1           ;SET SCOPE PTR
831 003564 012746 000001          MOV     #1,-(SP)        ;PUSH BANK # ON THE STACK
832 003570 010046          MOV     R0,-(SP)        ;PUSH # OF 256. WORD BLOCKS TO WRITE
833 003572 004767 002506          JSR     PC,..3X9        ;GO CHECK DATA WRITTEN
834
835          ;SETUP TO RUN MODIFIED 3 XOR 9 PATTERN IF PARITY MEMORY IS AVAILABLE
836 003576 022767 177776 175156  CMP     #-2,MMAVA
837 003604 001403          BEQ     1$
838 003606 005737 005464          TST     2#PARAVA        ;BRANCH IF PARITY MEMORY IS NOT AVAIL
839 003612 001406          BEQ     DONE8
840 003614 005737 003472          1$:    TST     2#PARPAT        ;BRANCH IF PARITY PAT JUST WRITTEN
    
```



```

841 003620 001003          BNE  DONE8
842 003622 010637 003472  MOV  SP, @PARPAT      ;SET INDICATOR TO WRITE 3X9 PAR PAT
843 003626 000730          BR   DONE8           ;REPEAT TEST USING MODIFIED 3X9 PATTERN
844
845          :WRITE 8 XOR 13 TEST PATTERN STARTING AT ADDRESS 40000
846 003630 012706 000500  DONE8: MOV  #STKPTR, SP      ;SET STACK PTR
847 003634 012737 003656 000004  MOV  #DONE9, @ERRVEC   ;SET TIME OUT TRAP VECTOR
848 003642 010701          MOV  PC, R1           ;UPDATE TRACE REGISTER
849 003644 012746 000002          MOV  #2, -(SP)        ;PUSH STARTING BANK # ON THE STACK
850 003650 005046          CLR  -(SP)           ;PUSH # OF BANKS TO WRITE ON THE STACK
851 003652 004767 003222          JSR  PC, .8X13       ;GO TO ROUTINE TO WRITE DATA
852
853          :CHECK 8 XOR 13 TEST PATTERN WRITTEN ABOVE
854 003656 012706 000500  DONE9: MOV  #STKPTR, SP      ;SET STACK PTR
855 003662 010701          MOV  PC, R1           ;UPDATE TRACE REGISTER
856 003664 012737 001126 000004  MOV  #ERRTRP, @ERRVEC
857 003672 012746 000002          MOV  #2, -(SP)
858 003676 005404          NEG  R4
859 003700 042704 000001          BIC  #1, R4          ;SET 4K BANK COUNT TO 8K INCREMENT
860 003704 001403          BEQ  DONE10         ;DO NOT CHECK IF ONLY 12K
861 003706 010446          MOV  R4, -(SP)
862 003710 004767 003172          JSR  PC, ..8X13     ;GO CHECK 8 XOR 13 PATTERN WRITTEN ABOVE
863
864
865 003714 000005          DONE10: RESET      ;DISABLE MEM MGMT AND PARITY ACTION
866
867
868          .SBTTL  RANDOM DATA ROTATING I/O TESTS
869          :RANDOM DATA TEST. THIS TEST MOVES THE PROGRAM CODE THROUGHOUT MEMORY
870 003716 010701 004056 000004  RANTST: MOV  PC, R1           ;SET TRACE POINTER
871 003720 012737 004056 000004  MOV  #7$, @ERRVEC     ;SET TIME OUT TRAP
872 003726 005767 175030          TST  MAVA           ;CHECK IF MEM MGMT IS AVAILABLE
873 003732 001412          BEQ  1$            ;BRANCH IF NOT AVAILABLE
874 003734 004767 001554          JSR  PC, LDMMO       ;GO SET UP MEM MGMT
875 003740 105237 172301          INCB @#KIPDR0+1     ;ALLOW 4K ADDRESSING IN FIRST 4K
876 003744 012737 077406 172304  MOV  #200*256.-400+UP, @#KIPDR2 ;SET KIPDR2=RW UP 200 BLOCKS
877 003752 012737 000400 172344  MOV  #400, @#KIPDR2
878 003760 012702 020000 1$:  MOV  #20000, R2      ;SET 'TO' ADDRESS POINTER
879 003764 005004          CLR  R4            ;SET 'FROM' ADDRESS POINTER
880 003766 012705 004000 2$:  MOV  #2048, R5       ;SET 4K WORD COUNT
881 003772 012422 3$:  MOV  (R4)+, (R2)+   ;MOVE CODE
882 003774 012422          MOV  (R4)+, (R2)+
883 003776 005305          DEC  R5            ;DECREMENT 4K WORD COUNTER
884 004000 001374          BNE  3$
885
886 004002 012705 005405          MOV  #4096.-PLACE+1, R5 ;SET 4K WORD COUNTER
887 004006 014400 4$:  MOV  -(R4), R0      ;GET 'GOOD' DATA
888 004010 014203          MOV  -(R2), R3     ;GET 'BAD' DATA
889 004012 020003          CMP  R0, R3        ;COMPARE 'GOOD' & 'BAD' DATA
890 004014 001403          BEQ  5$
891 004016 005722          TST  (R2)+         ;STEP ADDRESS FOR ERROR ROUTINE
892 004020 104400          HLT                ;REPORT ERROR
893 004022 005742          TST  -(R2)        ;RESTORE ADDRESS POINTER
894 004024 005305 5$:  DEC  R5            ;DECREMENT 4K WORD COUNTER
895 004026 001367          BNE  4$            ;LOOP UNTIL 4K WORDS CHECKED
896

```

```

897 004030 005767 174726      TST      MAVA          ;CHECK IF MEM MGMT IS AVAILABLE
898 004034 001405              BEQ      6$           ;BRANCH IF NOT AVAILABLE
899 004036 005237 172342      INC      @#KIPAR1
900 004042 005237 172344      INC      @#KIPAR2
901 004046 000744              BR       1$
902 004050 062702 000100      6$: ADD     #64.,R2      ;STEP ADDRESS
903 004054 000744              BR       2$
904 004056 012706 000500      7$: MOV     #STKPTR,SP   ;RESET STACK PTR
905 004062 012737 001126 000004 MOV     #ERRTRP,@#ERRVEC;RESTORE ERROR TRAP VECTOR
906
907      ;ROTATING 0 TEST. THIS TEST ROTATES A SINGLE '0' THROUGH MEMORY
908 004070 012767 177777 003264 ROTO: MOV     #-1,.CONST ;SET CONSTANT =177777
909 004076 012746 000001      MOV     #1,-(SP)       ;SET BANK #1
910 004102 016746 177446      MOV     WDS.256,-(SP)  ;GET # OF 256. WORD BLOCKS IN MEMORY
911 004106 004767 003232      JSR     PC,WRTPAT      ;GO WRITE 1'S THROUGHOUT MEMORY
912 004112 010701              MOV     PC,R1          ;SET SCOPE PTR
913 004114 012746 000001      MOV     #1,-(SP)       ;SET STARTING BANK #
914 004120 016746 177430      MOV     WDS.256,-(SP)  ;SET # OF 256. WORD BLOCKS TO CHECK
915 004124 004767 002764      JSR     PC,.ROTO       ;GO TO ROTATE 0 ROUTINE
916
917      ;ROTATING 1 TEST THIS TEST ROTATES A SINGLE '1' BIT THROUGH ALL OF
918      ;MEMORY
919 004130 005067 003226      ROT1: CLR     .CONST    ;CLEAR CONSTANT
920 004134 012746 000001      MOV     #1,-(SP)       ;PUSH STARTING BANK ONTO STACK
921 004140 016746 177410      MOV     WDS.256,-(SP)  ;AND # OF 256. WORD BLOCKS IN MEMORY
922 004144 004767 003174      JSR     PC,WRTPAT      ;GO WRITE 0'S THROUGHOUT MEMORY
923 004150 010701              MOV     PC,R1          ;SET SCOPE PTR
924 004152 012746 000001      MOV     #1,-(SP)       ;SET STARTING BANK #
925 004156 016746 177372      MOV     WDS.256,-(SP)  ;SET # OF 256. WORD BLOCKS TO CHECK
926 004162 004767 003022      JSR     PC,.ROT1       ;GO ROTATE A '1' BIT THROUGHOUT MEMORY
927
928      ;END OF CYCLE
929 004166 000005      END:  RESET
930 004170 042737 000014 177746 BIC     #14,@#CNTRL    ;RESET MACHINE TO KEY-START STATE
931 004176 010701              MOV     PC,R1          ;UPDATE TRACE REGISTER
932 004200 012706 000500      MOV     #STKPTR,SP    ;SET STACK PTR
933 004204 005237 000752      INC     @#ICNT         ;INCREMENT PASS COUNT
934 004210 022737              CMP     (PC)+,@(PC)+  ;CHECK FOR LAST PASS
935 004212 000006      ENDCT: .WORD    6       ;MAKE 5 PASSES
936 004214 000752              .WORD    ICNT         ;PASS COUNT ADDRESS
937 004216 001405              BEQ     DONE          ;BRANCH IF LAST PASS COMPLETED
938 004220 004567 174544      JSR     R5,$PRINT     ;GO TO PRINT ROUTINE
939 004224 010106      ASTERISK
940 004226 000137 003446      JMP     @#BEGIN1
941
942 004232 004567 174532      DONE: JSR     R5,$PRINT     ;GO TO PRINT ROUTINE
943 004236 010110      ENDMSG
944 004240 105737 177564      TSTB   @#TPS          ;WAIT FOR BELL TO RING
945 004244 100375              BPL     .-4
946 004246 013700 000042      MOV     @#42,R0       ;GET DECTAPE MONITOR RETURN ADDRESS
947 004252 001406              BEQ     FINISH
948 004254 004767 175730      JSR     PC,$RLDR      ;RESTORE MONITOR & LOADERS
949 004260 004710      SENDAD: JSR     PC,(R0)   ;GO TO DECTAPE MONITOR
950 004262 000240              NOP
951 004264 000240              NOP
952 004266 000240              NOP

```

```

953 004270 000167 176144 FINISH: JMP START1
954
955 .SBTTL PROGRAM SUBROUTINES
956 .SBTTL RELOCATION ROUTINES
957 ;ROUTINE TO RELOCATE PROGRAM CODE
958 004274 012500 RELOC: MOV (R5)+,R0 ;GET FROM ADDRESS
959 004276 011502 MOV (R5),R2 ;GET TO ADDRESS
960 004300 010203 MOV R2,R3
961 004302 062703 017776 000004 ADD #17776,R3 ;MOVES 4K
962 004306 012737 004356 000004 MOV #45,2#ERRVEC ;SET TIME OUT TRAP
963 004314 005004 CLR R4 ;CLEAR RELOCATION SUCCESSFUL INDICATOR
964 004316 005723 TST (R3)+ ;CHECK IF MEMORY IS AVAILABLE
965 004320 012022 1$: MOV (R0)+,(R2)+ ;RELOCATE
966 004322 020203 CMP R2,R3 ;RELOCATION COMPLETE?
967 004324 001375 BNE 1$
968 004326 011503 MOV (R5),R3
969 004330 020203 2$: CMP R2,R3
970 004332 001413 BEQ 5$ ;BRANCH IF DONE
971 004334 024042 CMP -(R0),-(R2) ;CHECK THAT DATA WAS RELOCATED PROPERLY
972 004336 001774 BEQ 2$
973 004340 005703 TST R3 ;CHECK IF RELOCATING BACK TO 000000
974 004342 001403 BEQ 3$
975 004344 104400 HLT ;ERROR! CANNOT RELOCATE PROGRAM CODE
976 ;TO UPPER MEMORY BANK PROPERLY
977 004346 000000 HALT
978 004350 000767 BR 2$ ;CONTINUE RELOCATING AT YOUR PERIL
979 004352 000000 3$: HALT ;ERROR! CANNOT RELOCATE CODE BACK TO
980 ;TO 000000 PROPERLY
981 004354 000777 BR
982 004356 022626 4$: CMP (SP)+,(SP)+ ;RESTORE STACK PTR
983 004360 005104 COM R4
984 004362 000240 5$: NOP
985 004364 012702 000764 MOV #RELOC,R2 ;GET ADDRESS OF RELOCATION FACTOR
986 004370 061502 ADD (R5),R2 ;ADD FACTOR
987 004372 012512 MOV (R5)+,(R2) ;RELOCATED RELOC NOW CONTAINS RELOCATION
988 ;FACTOR
989 004374 000205 RTS 5 ;RETURN, R4=-1 IF NO RELOCATION
990
991
992 ;ROUTINE TO RELOCATE PROGRAM CODE FROM ORIGINAL POSITION (0-4K) TO
993 ;TOP OF MEMORY.
994 004376 012700 020000 000004 RELOCP: MOV #20000,R0 ;SET UP TO SCAN FOR TOP OF MEMORY
995 004402 012737 000006 000004 MOV #ERRVEC+2,2#ERRVEC
996 004410 062700 020000 1$: ADD #20000,R0 ;INCREMENT SCAN ADDRESS
997 004414 000261 SEC ;SET TIME OUT INDICATOR
998 004416 005710 TST (R0) ;CHECK FOR EXISTANT MEMORY
999 004420 103373 BCC 1$ ;'C' WILL BE CLEAR IF MEMORY EXISTS
1000 004422 012737 001126 000004 MOV #ERRTRP,2#ERRVEC
1001 004430 162700 020000 SUB #20000,R0 ;ADJUST TO LAST EXISTANT 4K
1002 004434 010067 000006 MOV R0,2$ ;PASS RELOCATION ADDRESS TO RELOC ROUTINE
1003 004440 004567 177630 JSR RS,RELOC ;RELOCATE PROGRAM
1004 004444 000000 ;FROM ADDRESS 000000
1005 004446 000000 2$: .WORD 0 ;TO LAST 4K BANK
1006 004450 004567 174314 JSR RS,$PRINT ;GO TO PRINT ROUTINE
1007 004454 010047 RELOCM
1008 004456 016746 177764 MOV 2$,-(SP) ;PASS TO 02A ROUTINE

```

```

1009 004462 062716 010124      ADD    #REL24K,(SP)    ;SET UP RESTART ADDRESS
1010 004466 004767 175322      JSR    PC,02A         ;TYPE RESTART ADDRESS
1011 004472 011667 000006      MOV    (SP),3$       ;SAVE RETURN ADDRESS IN 3$ BELOW
1012 004476 066706 177744      ADD    2$,SP         ;RESET STACK PTR
1013 004502 012716              MOV    (PC)+,(SP)    ;GET RETURN ADDRESS
1014 004504 000000              3$: .WORD 0          ;CONTAINS RETURN PC
1015 004506 066716 177734      ADD    2$,(SP)       ;ADJUST RETURN PC
1016 004512 000207              RTS    PC

;SHTTL MA/MF PARITY ERROR SERVICE ROUTINE
;WHEN MA/MF A PARITY ERROR IS DETECTED THIS ROUTINE SCANS MEMORY FOR THE
;ADDRESS CAUSING THE PARITY ERROR. WHEN THE ADDRESS IS LOCATED THE ROUTINE
;HALTS WITH THE ADDRESS+2 IN R0. TO CONTINUE AFTER THE ERROR PRESS CONTINUE.
1019 .PARSRV:MOV    R0,SAVRO    ;SAVE R0 IN SAVRO
1020      MOV    #SAVRO+2,R0
1021      MOV    R1,(R0)+
1022      MOV    R2,(R0)+
1023      MOV    R3,(R0)+
1024      MOV    R4,(R0)+
1025      MOV    R5,(R0)+
1026      JSR    R5,$PRINT    ;GO TO PRINT ROUTINE
1027      PARERR
1028      CLR    (PC)+        ;CLEAR PARITY ERROR INDICATORS
1029      PEFLG: .BYTE 0      ;NOT 0/0 =PAR ERR/NO PAR ERR
1030      PENFLG: .BYTE 0    ;NOT 0/0=PAR ERR DETECTED/NOT DETECTED ON SCAN
1031      MOV    #2$,@#PARVEC ;SET PARITY ERROR TRAP
1032      MOV    #4$,@#ERRVEC ;SET TIME OUT TRAP VECTOR
1033      CLR    R2
1034      TST    MMAVA        ;CHECK IF MEM MGMT IS AVAILABLE
1035      BEQ    1$          ;BRANCH IF NOT AVAILABLE
1036      JSR    PC,LDMMO     ;SET UP MEM MGMT
1037      INCB  @#KIPDRO+1   ;ALLOW FULL 4K PAGE ADDRESSING
1038      MOV    #MMA8TO,@#MMVEC ;SET MEM MGMT ABORT TRAP VECTOR
1039      1$: MOV    (R2)+,R0 ;SCAN ALL ADDRESSES
1040      BR    1$
1041      2$: MOV    SP,PEFLG ;SET PARITY ERROR FOUND INDICATOR
1042      MOV    R0,R3
1043      HLT                    ;PARITY ERROR! ADDRESS+2 IS IN R2
1044      RTI                    ;CONTINUE SCAN
1045      3$: NOP                ;INSERT HALT INST TO EXAMINE PARITY REGS
1046      CLR    PEFLG         ;CLEAR PARITY ERROR INDICATORS
1047      MOV    #STKPTR,SP    ;RESET STACK PTR
1048      RESET
1049      JSR    PC,MAMF       ;GO ENABLE PARITY ERROR DETECTION
1050      JMP    @PERSTAT      ;RESTART SELECTED PROGRAM

;SERVICE ROUTINE IF PARITY ERROR NOT DETECTED ON SCAN
1051 4$: TSTB PEFLG           ;BRANCH IF PARITY ERROR WAS
1052      BNE  3$             ;DETECTED ON SCAN
1053      MOV    4(SP),R2     ;GET PC AT TIME OF ERROR
1054      SUB    #2,R2        ;BACK IT UP
1055      MOV    SP,PENFLG    ;SET IND = NO PAR ERROR DETECTED ON SCAN
1056      JSR    R5,$PRINT    ;GO TO PRINT ROUTINE
1057      HLT                    ;ERROR! PARITY ERROR NOT DETECTED ON SCAN
1058      BR    3$

```

```

1065 ;THE BELOW 6 WORDS CONTAINS THE SAVED CONTENTS OF R0-R5 WHEN THE
1066 ;PARITY ERROR OCCURRED
1067 004710 000000 SAVR0: .WORD 0
1068 004712 000000 SAVR1: .WORD 0
1069 004714 000000 SAVR2: .WORD 0
1070 004716 000000 SAVR3: .WORD 0
1071 004720 000000 SAVR4: .WORD 0
1072 004722 000000 SAVR5: .WORD 0
1073
1074 004724 005015 040520 044522 PARERR: .ASCIZ <15><12>'PARITY ERROR'<15><12>
1075 004732 054524 042140 051122
1076 004740 051117 005015 000
1077 004745 116 052117 043040 NOFIND: .ASCIZ 'NOT FOUND ON SCAN'<15><12>
1078 004752 052517 042116 047440
1079 004760 020116 041523 047101
1080 004766 005015 000
1081 .EVEN
1082
1083
1084 177740 MEMLO=177740
1085 177742 MEMHI=177742
1086 177744 MEMERR=177744
1087
1088
1089 004772 005767 177550 .22PAR: TST PEFLG ;BEEN HERE BEFORE
1090 004776 001403 BFC 15 ;BRANCH IF NO
1091 005000 000000 HALT ;YES -- DOUBLE PARITY ERROR
1092 005002 000177 175454 JMP 2PERSTR
1093 005006 010667 177534 15: MOV SP,PEFLG ;SET PARITY ERROR FLAG
1094 005012 005737 177570 TST 2#SWR ;HALT ON ERROR?
1095 005016 100001 BPL 100$ ;BRANCH IF NO
1096 005020 000000 HALT ;YES
1097 005022 013746 177744 100$: MOV 2#MEMERR,-(SP) ;SAVE MEMORY ERROR REG
1098 005026 013701 177740 MOV 2#MEMLO,R1 ;GET ADDRESS OF WHERE THE PARITY
1099 005032 013702 177742 MOV 2#MEMHI,R2 ;ERROR OCCURRED
1100 005036 011637 177744 MOV (SP),2#MEMERR ;CLEAR THE ERROR REG
1101 005042 032737 020000 177570 BIT #BIT13,2#SWR ;INHIBIT ERROR TYPEOUT
1102 005050 001071 BNE 101$ ;BRANCH IF YES
1103 ;PRINT "PARITY ERROR"
1104 005052 004567 173712 JSR R5,$PRINT
1105 005056 004724 PARERR
1106 ;PRINT "PC=XXXXXX"
1107 005060 004567 173704 JSR R5,$PRINT
1108 005064 001475 ERRPC
1109 005066 016646 000002 MOV 2(SP),-(SP) ;GET PC AT TIME OF PARITY ERROR
1110 005072 066716 173666 ADD RELOC,(SP)
1111 005076 004767 174712 JSR PC,02A
1112 005102 004567 173662 JSR R5,$PRINT
1113 005106 002361 SPACE1
1114 ;CHANGE 22-BIT ADDRESS TO OCTAL-ASCII
1115 005110 012700 002351 MOV #DIGITS,R0
1116 005114 012704 000010 MOV #8,R4
1117 005120 012705 000003 2$: MOV #3,R5
1118 005124 005003 3$: CLR R3
1119 005126 006301 4$: ASL R1
1120 005130 106102 ROLB R2

```

```

1121 005132 006103          ROL      R3
1122 005134 077504          SOB      R5,4$
1123 005136 116320 002312  MOVB    DIGTAB(R3),(R0)+
1124 005142 077412          SOB      R4,2$
1125          ;PRINT  "MEMORY ADDRESS IS AAAAAAA"
1126 005144 004567 173620  JSR     RS,$PRINT
1127 005150 002327          ADRSS
1128 005152 004767 172722  JSR     PC,CRLF
1129          ;PRINT  "PARITY ERROR REG=XXXXXX"
1130 005156 004567 173606  JSR     RS,$PRINT
1131 005162 001531          PARREG
1132 005164 011605          MOV     (SP),R5
1133 005166 004767 174622  JSR     PC,02A
1134 005172 004767 173572  JSR     PC,$PRINT
1135 005176 002361          SPACE1
1136          ;PRINT  THE MARGIN SETTING
1137 005200 016700 173546  MOV     ICNT,R0
1138 005204 116000 005506  MOVB    MARGTB(R0),R0
1139 005210 062700 005240  ADD     #MARTBL,R0
1140 005214 011067 000004  MOV     (R0),5$
1141 005220 004567 173544  JSR     RS,$PRINT
1142 005224 005240          SS:    MARTBL
1143 005226 004567 173536  JSR     RS,$PRINT
1144 005232 005347          MARMMSG
1145 005234 000177 175222  1015:  JMP     @PERSTR
1146          ;MARGIN MESSAGE TABLE
1147 005240 005256          MARTBL: NORMAL
1148 005242 000000          0
1149 005244 005265          ESTRB
1150 005246 005302          LSTRB
1151 005250 005316          LCRNT
1152 005252 005332          HCRNT
1153 005254 005256          NORMAL
1154
1155          ;MARGIN MESSAGES
1156 005256 047516 046522 046101  NORMAL: .ASCIZ 'NORMAL'
1157 005264          000
1158 005265          105 051101 054514  ESTRB: .ASCIZ 'EARLY STROBE'
1159 005272 051440 051124 041117
1160 005300 000105
1161 005302 040514 042524 051440  LSTRB: .ASCIZ 'LATE STROBE'
1162 005310 051124 041117 000105
1163 005316 047514 020127 052503  LCRNT: .ASCIZ 'LOW CURRENT'
1164 005324 051122 047105 000124
1165 005332 044510 044107 041440  HCRNT: .ASCIZ 'HIGH CURRENT'
1166 005340 051125 042522 052116
1167 005346          000
1168 005347          040 040515 043522  MARMMSG: .ASCIZ ' MARGIN'<12><15>
1169 005354 047111 006412          000
1170          .EVEN
1171
1172          ;ROUTINE TO ENABLE PARITY ERROR ACTION ON 11/70 PARITY MEMORIES
1173          000114          PARVEC=114          ;PARITY ERROR INTERRUPT VECTOR ADDRESS
1174
1175 005362 032737 000040 177570  .MAMF:  BIT     #40,@SWR          ;CHECK IF PARITY ERROR DETECTION IS TO
1176 005370 001007          BNE     1$          ;BE ENABLED. BRANCH IF NOT TO BE ENABLED
    
```

```

1177 ;ENABLE PARITY ERROR DETECTION
1178 005372 042737 000002 177746 BIC #2,2#CNTRL ;OTHERWISE, INSURE THAT PARITY ERROR
1179 ;DETECTION IS ENABLED
1180 005400 012767 000001 000056 MOV #1,PARAVA ;SET PARITY ERROR DETECTION INDICATOR
1181 005406 000405 BR 2$
1182 ;DISABLE PARITY ERROR DETECTION
1183 005410 052737 000002 177746 1$: BIS #2,2#CNTRL ;DISABLE PARITY ERROR DETECTION
1184 005416 005067 000042 CLR PARAVA ;CLEAR PARITY ERROR DETECTION INDICATOR
1185 ;SET-UP PARITY ERROR SERVICE TRAP FOR 18-BIT OR 22-BIT
1186 ;ADDRESSING MODES
1187 005422 012737 004514 000114 2$: MOV #.PARSRV,2#PARVEC ;SET-UP 18-BIT ADDRESS PARITY
1188 ;ERROR TRAP VECTOR
1189 005430 012737 000340 000116 MOV #340,2#PARVEC+2 ;PRIORITY LEVEL 7 ON TRAP
1190 005436 022767 177776 173316 CMP #-2,MMAVA ;22-BIT ADDRESSING ENABLED?
1191 005444 001006 BNE 3$ ;BRANCH IF NOT, OTHERWISE
1192 005446 012737 004772 000114 MOV #.22PAR,2#PARVEC ;SET-UP 22-BIT ADDRESS PARITY
1193 ;ERROR TRAP VECTOR
1194 005454 012737 000340 000116 MOV #340,2#PARVEC+2 ;PRIORITY LEVEL 7 ON TRAP
1195 005462 000207 3$: RTS PC ;RETURN
1196 005464 000000 PARAVA: .WORD 0 ;PARITY ERROR DETECTION INDICATOR
1197 ;0 - PARITY ERROR DETECTION IS DISABLED
1198 ;1 - PARITY ERROR DETECTION IS ENABLED
1199
1200
1201 ;.SBTTL MARGIN ROUTINE
1202 ;ROUTINE TO SET THE MARGINS
1203 177750 MAINTRG=177750
1204
1205 005466 016700 173260 MARGIN: MOV ICNT,RC ;PASS COUNT
1206 005472 005002 CLR R2 ;FAST COUNTER
1207 005474 116037 005506 177750 MOV#B MRGNTB(RO),2#MAINTRG ;LOAD MAINTENANCE REG.
1208 005502 077201 1$: SOB R2,1$
1209 005504 000207 RTS PC
1210
1211 005506 000 MRGNTB: .BYTE 0 ;NORMAL
1212 005507 004 .BYTE 4 ;EARLY STROBE
1213 005510 006 .BYTE 6 ;LATE STROBE
1214 005511 010 .BYTE 10 ;LOW CURRENT
1215 005512 012 .BYTE 12 ;HIGH CURRENT
1216 005513 000 .BYTE 0 ;NORMAL
1217
1218
1219 ;.SBTTL MEM MGMT ROUTINES
1220 ;ROUTINE TO INITIALIZE MEMORY MANAGEMENT REGISTERS
1221 005514 000240 LDMMO: NOP
1222 005516 005767 173240 TST MMAVA
1223 005522 001434 BEQ 1$
1224 005524 012737 000020 172516 MOV #20,2#SR3 ;22 BIT MODE
1225 005532 012737 077006 172300 MOV #177*256.-400+UP+RW,2#KIPDR0 ;SET KIPDR0=RW UP 177 BLOCKS
1226 005540 012737 077406 172302 MOV #200*256.-400+UP+RW,2#KIPDR1 ;SET KIPDR1=RW UP 200 BLOCKS
1227 005546 005037 172304 CLR 2#KIPDR2
1228 005552 005037 172344 CLR 2#KIPDR2
1229 005556 012737 077406 172316 MOV #200*256.-400+UP+RW,2#KIPDR7 ;SET KIPDR7=RW UP 200 BLOCKS
1230 005564 005037 172340 CLR 2#KIPDR0
1231 005570 012737 000200 172342 MOV #200,2#KIPDR1
1232 005576 012737 177600 172356 MOV #177600,2#KIPDR7

```



```

1233 005604 012737 000001 177572      MOV      #1, @#SR0      ;ENABLE MEM MGMT
1234 005612 000240                      NOP
1235 005614 000207      1$:     RTS      PC
1236
1237      ;MEMORY MANAGEMENT ABORT ROUTINE FOR WRITE UP
1238 005616 012702 020000      MMABT0: MOV      #20000, R2      ;RESET R2
1239 005622 062737 000200 172342      ADD      #200, @#KIPAR1      ;ADVANCE TO NEXT 4K
1240 005630 013716 177576      MOV      @#SR2, (SP)      ;RETURN TO INSTRUCTION THAT
1241 005634 005037 177572      CLR      @#SR0      ;DISABLE MEM MGMT
1242 005640 012737 000001 177572      MOV      #1, @#SR0      ;ENABLE MEM MGMT
1243 005646 000002                      RTI      ;CAUSED THE ABORT
1244
1245      ;MEM MGMT ABORT SERVICE FOR WRITE DOWN
1246 005650 012702 040000      MMABT1: MOV      #40000, R2      ;RESET R2
1247 005654 162737 000200 172342      SUB      #200, @#KIPAR1
1248 005662 001406                      BEQ      2$
1249 005664 013716 177576      MOV      @#SR2, (SP)
1250 005670 012737 000001 177572      MOV      #1, @#SR0      ;ENABLE MEM MGMT
1251 005676 000002                      RTI
1252
1253 005700 005037 177572      2$:     CLR      @#SR0      ;DISABLE MEM MGMT
1254 005704 052766 000002 000002      BIS      #V, 2(SP)
1255 005712 000002                      RTI
1256
1257      ;ROUTINE TO SET UP MEMORY MANAGEMENT FOR PATTERN TESTS
1258 005714 005702      STMM2: TST      R2      ;CHECK IF TESTING BANK # 0
1259 005716 001442                      BEQ      2$      ;EXIT IF BANK # 0
1260 005720 005767 173036      TST      MMVA
1261 005724 001005                      BNE      1$      ;BRANCH IF MEM MGMT AVAILABLE
1262 005726 006002                      ROR      R2      ;ADJUST ADDRESS
1263 005730 006002                      ROR      R2
1264 005732 006002                      ROR      R2
1265 005734 006002                      ROR      R2
1266 005736 000207                      RTS      PC      ;RETURN
1267
1268 005740 004767 177550      1$:     JSR      PC, LDMMO      ;GO MAKE INITIAL SET UP
1269 005744 000302                      SWAB    R2
1270 005746 006002                      ROR      R2
1271 005750 010237 172344      MOV      R2, @#KIPAR2
1272 005754 062702 000200      ADD      #200, R2
1273 005760 010237 172346      MOV      R2, @#KIPAR3
1274 005764 012737 077406 172304      MOV      #200*256.-400+UP+RW, @#KIPDR2      ;SET KIPDR2=RW UP 200 BLOCKS
1275 005772 012737 077406 172306      MOV      #200*256.-400+UP+RW, @#KIPDR3      ;SET KIPDR3=RW UP 200 BLOCKS
1276 006000 005037 172310      CLR      @#KIPDR4
1277 006004 012702 040000      MOV      #40000, R2
1278 006010 012737 006026 000250      MOV      @MMABT2, @MMVEC
1279 006016 012737 000001 177572      MOV      #1, @#SR0      ;ENABLE MEM MGMT
1280 006024 000207      2$:     RTS      PC
1281
1282      ;ROUTINE TO SERVICE 8 XOR 13 ABORTS
1283 006026 000240      MMABT2: NOP
1284 006030 012702 040000      MOV      #40000, R2
1285 006034 062737 000400 172344      ADD      #400, @#KIPAR2
1286 006042 062737 000400 172346      ADD      #400, @#KIPAR3
1287 006050 013716 177576      MOV      @#SR2, (SP)      ;SET RETURN TO INSTRUCTION THAT ABORTED
1288 006054 012737 000001 177572      MOV      #1, @#SR0      ;ENABLE MEM MGMT
    
```

```

1289 006062 000002 RTI
1290
1291
1292
1293
1294
1295
1296
1297 005064 016602 000004 .3X9: MOV 4(SP),R2 ;GET STARTING BANK #
1298 006070 004767 177620 JSR PC,STMM2
1299 006074 005000 CLR R0
1300 006076 010003 MOV R0,R3
1301 006100 005103 COM R3 ;R0 (0) AND R3 (-1) IS THE DATA WRITTEN
1302 006102 005767 175364 TST PARPAT ;BRANCH IF PARITY MEMORY PATTERN IS
1303 006106 001402 BEQ 1$ ;NOT TO BE WRITTEN
1304
1305 006110 012700 00040' 1$: MOV #401,R0 ;WRITE PARITY 3X9 PATTERN
1306 006114 012704 00002L MOV #16.,R4 ;EACH LOOP WRITES 256. WORDS
1307
1308 006120 010022 2$: MOV R0,(R2)+
1309 006122 010022 MOV R0,(R2)+
1310 006124 010022 MOV R0,(R2)+
1311 006126 010022 MOV R0,(R2)+
1312
1313 006130 010022 MOV R0,(R2)+
1314 006132 010022 MOV R0,(R2)+
1315 006134 010022 MOV R0,(R2)+
1316 006136 010022 MOV R0,(R2)+
1317
1318 006140 010322 MOV R3,(R2)+
1319 006142 010322 MOV R3,(R2)+
1320 006144 010322 MOV R3,(R2)+
1321 006146 010322 MOV R3,(R2)+
1322
1323 006150 010322 MOV R3,(R2)+
1324 006152 010322 MOV R3,(R2)+
1325 006154 010322 MOV R3,(R2)+
1326 006156 010322 MOV R3,(R2)+
1327
1328 006160 005304 DEC R4
1329 006162 001356 BNE 2$
1330 006164 005100 COM R0
1331 006166 005103 COM R3
1332 006170 005767 175276 TST PARPAT ;BRANCH IF PARITY MEMORY PATTERN IS
1333 006174 001402 BEQ 3$ ;NOT TO BE WRITTEN
1334
1335 006176 004767 000014 3$: JSR PC,XOR39 ;GO GET CONSTANTS
1336 006202 005366 000002 DEC 2(SP) ;DECREMENT 256. WORD BLOCK COUNT
1337 006206 001342 BNE 1$
1338 006210 012616 MOV (SP)+,(SP) ;ADJUST STACK
1339 006212 012616 MOV (SP)+,(SP)
1340 006214 000207 RTS PC
1341
1342 ;ROUTINE TO SET CONSTANTS FOR WRITING/CHECKING 3 XOR PATTERN WITH
1343 ;PARITY.
1344 006216 032702 000020 .XOR39: BIT #20,R2 ;CHECK BIT 3

```

```

1345 006222 001404          BEQ      .3IS0          ;BRANCH IF BIT 3 = 0
1346 006224 032702 002000 .3IS1: BIT      #2000,R2    ;CHECK BIT 9
1347 006230 001404          BEQ      .3NOT9         ;BRANCH IF BIT 9 =0
1348 006232 000407          BR       .3IS9
1349 006234 032702 002000 .3IS0: BIT      #2000,R2    ;CHECK BIT 9
1350 006240 001404          BEQ      .3IS9         ;BRANCH IF 0
1351 006242 005767 172506 .3NOT9: TST     ICOUNT     ;CHECK IF NORMAL OR COMPLEMENT DATA
1352 006246 100004          BPL     LDCOMP         ;GO LOAD COMPLEMENT CONSTANTS
1353 006250 100410          BMI     LDNORM        ;GO LOAD NORMAL CONSTANTS
1354 006252 005767 172476 .3IS9: TST     ICOUNT     ;CHECK IF NORMAL OR COMPLEMENT DATA
1355 006256 100005          BPL     LDNORM        ;GO LOAD NORMAL CONSTANTS
1356 006260 012700 177777 LDCOMP: MOV     #-1,R0     ;SET COMPLEMENT CONSTANTS
1357 006264 012703 000401      MOV     #401,R3
1358 006270 000207          RTS     PC             ;RETURN
1359 006272 012700 000401 LDNORM: MOV     #401,R0     ;LOAD NORMAL CONSTANTS
1360 006276 012703 177777      MOV     #-1,R3
1361 006302 000207          RTS     PC
1362
1363      ;ROUTINE TO CHECK 3 XOR 9 WORST CASE NOISE PATTERN
1364      ;CALL: MOV     BANK#,-(SP) ;PUSH STARTING BANK # ONTO STACK
1365      ;      MOV     BLKCNT,-(SP) ;AND 256. WORD BLOCK COUNT
1366      ;      JSR     PC,..3X9     ;CALL ROUTINE
1367
1368 006304 000240          ..3X9: NOP
1369 006306 004767 001104      JSR     PC,CKSWR      ;GO CHECK SWITCH REGISTER
1370
1371      ;CHECK WORST CASE PATTERN
1372 006312 016604 000002 1$:  MOV     2(SP),R4     ;GET 256. BLOCK WORD COUNT
1373 006316 016602 000004      MOV     4(SP),R2     ;GET FIRST BANK #
1374 006322 004767 177366      JSR     PC,STMM2     ;GO SET UP MEM MGMT
1375 006326 005000          CLR     R0           ;SET CHECK WORD
1376 006332 005767 172420      TST     ICOUNT     ;IF ICOUNT IS NEG AM CHECKING COMP-
1377 006334 100001          BPL     .+4         ;PLEMENTED PATTERN
1378 006336 005100          COM     R0           ;SO COMPLEMENT CHECK WORD
1379 006340 012705 000040 2$:  MOV     #32.,R5     ;SET 256. WORD COUNTER
1380
1381 006344 005767 175122 3$:  TST     PARPAT
1382 006350 001402          BEQ     30$
1383
1384 006352 004767 177640          JSR     PC,.XOR39    ;GO GET CONSTANT
1385 006356
1386 006356 012203          30$:  MOV     (R2)+,R3     ;GET TEST DATA
1387 006360 020003          CMP     R0,R3       ;COMPARE WITH CHECK WORD
1388 006362 001403          BEQ     .+10
1389 006364 005046          CLR     -(SP)       ;PUSH FAKE STATUS ON THE STACK
1390 006366 004767 172612      JSR     PC,ERROR     ;ERROR! MEM DATA (R3) NOT = TEST DATA
1391      ;(R0), ADDRESS=(R2)-2
1392
1393      ;GET TEST DATA
1394 006372 012203          MOV     (R2)+,R3     ;COMPARE WITH CHECK WORD
1395 006374 020003          CMP     R0,R3
1396 006376 001403          BEQ     .+10
1397 006400 005046          CLR     -(SP)       ;PUSH FAKE STATUS ON THE STACK
1398 006402 004767 172576      JSR     PC,ERROR     ;ERROR! MEM DATA (R3) NOT = TEST DATA
1399      ;(R0), ADDRESS=(R2)-2
1400 006406 012203          MOV     (R2)+,R3     ;GET TEST DATA

```

```

1401 006410 020003          CMP      R0,R3          ;COMPARE WITH CHECK WORD
1402 006412 001403          BEQ      .+10
1403 006414 005046          CLR      -(SP)         ;PUSH FAKE STATUS ON THE STACK
1404 006416 004767 172562        JSR      PC,ERROR      ;ERROR! MEM DATA (R3) NOT = TEST DATA
1405                                     ;(R0), ADDRESS=(R2)-2
1406
1407 006422 012203          MOV      (R2)+,R3      ;GET TEST DATA
1408 006424 020003          CMP      R0,R3          ;COMPARE WITH CHECK WORD
1409 006426 001403          BEQ      .+10
1410 006430 005046          CLR      -(SP)         ;PUSH FAKE STATUS ON THE STACK
1411 006432 004767 172546        JSR      PC,ERROR      ;ERROR! MEM DATA (R3) NOT = TEST DATA
1412                                     ;(R0), ADDRESS=(R2)-2
1413
1414 006436 012203          MOV      (R2)+,R3      ;GET TEST DATA
1415 006440 020003          CMP      R0,R3          ;COMPARE WITH CHECK WORD
1416 006442 001403          BEQ      .+10
1417 006444 005046          CLR      -(SP)         ;PUSH FAKE STATUS ON THE STACK
1418 006446 004767 172532        JSR      PC,ERROR      ;ERROR! MEM DATA (R3) NOT = TEST DATA
1419                                     ;(R0), ADDRESS=(R2)-2
1420
1421 006452 012203          MOV      (R2)+,R3      ;GET TEST DATA
1422 006454 020003          CMP      R0,R3          ;COMPARE WITH CHECK WORD
1423 006456 001403          BEQ      .+10
1424 006460 005046          CLR      -(SP)         ;PUSH FAKE STATUS ON THE STACK
1425 006462 004767 172516        JSR      PC,ERROR      ;ERROR! MEM DATA (R3) NOT = TEST DATA
1426                                     ;(R0), ADDRESS=(R2)-2
1427
1428 006466 012203          MOV      (R2)+,R3      ;GET TEST DATA
1429 006470 020003          CMP      R0,R3          ;COMPARE WITH CHECK WORD
1430 006472 001403          BEQ      .+10
1431 006474 005046          CLR      -(SP)         ;PUSH FAKE STATUS ON THE STACK
1432 006476 004767 172502        JSR      PC,ERROR      ;ERROR! MEM DATA (R3) NOT = TEST DATA
1433                                     ;(R0), ADDRESS=(R2)-2
1434
1435 006502 012203          MOV      (R2)+,R3      ;GET TEST DATA
1436 006504 020003          CMP      R0,R3          ;COMPARE WITH CHECK WORD
1437 006506 001403          BEQ      .+10
1438 006510 005046          CLR      -(SP)         ;PUSH FAKE STATUS ON THE STACK
1439 006512 004767 172466        JSR      PC,ERROR      ;ERROR! MEM DATA (R3) NOT = TEST DATA
1440                                     ;(R0), ADDRESS=(R2)-2
1441
1442
1443 006516 005100          COM      R0             ;COMPLEMENT CHECK WORD
1444 006520 005305          DEC      R5             ;DECREMENT 256. WORD COUNTER
1445 006522 001310          BNE
1446 006524 005100          COM      R0             ;COMPLEMENT CHECK WORD
1447 006526 005304          DEC      R4             ;DECREMENT BLOCK COUNTER
1448 006530 001303          BNE
1449
1450 006532 032737 040000 177570        BIT      #40000,2#SWR   ;LOOP ON TEST?
1451 006540 001264          BNE      1$            ;BRANCH IF LOOP ON TEST DESIRED
1452 006542 016667 000002 172216 40$:   MOV      2(SP),COUNT  ;GET # OF 256. WORD BLOCKS TO CHECK
1453 006550 016602 000004          MOV      4(SP),R2      ;GET STARTING BANK #
1454 006554 004767 177134          JSR      PC,STMM2      ;GO SET UP MEM MGMT IF REQUIRD
1455
1456                                     ;CHECK WORST CASE BIT COMPLEMENT PATTERN

```

1457	006560	005000		CLR	R0	
1458	006562	005767	172166	TST	ICOUNT	; CHECK IF COMPLEMENT PATTERN
1459	006566	100001		BPL	+4	
1460	006570	005100		COM	R0	; COMPLEMENT CHECK WORD
1461	006572	012704	000040	4\$: MOV	#32, R4	; SET 256. WORD COUNTER
1462	006576	012705	000010	5\$: MOV	#8, R5	; SET 8 WORD COUNTER
1463	006602	005767	174664	6\$: TST	PARPAT	; BRANCH IF PARITY MEMORY PATTERN IS
1464	006606	001402		BEQ	60\$; NOT TO BE CHECKED
1465	006610	004767	177402	JSR	PC, XOR39	
1466	006614	012203		60\$: MOV	(R2)+, R3	; GET DATA
1467	006616	020003		CMP	R0, R3	; CHECK DATA
1468	006620	001403		BEQ	+10	
1469	006622	005046		CLR	-(SP)	
1470	006624	004767	172354	JSR	PC, ERROR	
1471	006630	005100		COM	R0	; COMPLEMENT CHECK WORD
1472	006632	005142		COM	-(R2)	; COMPLEMENT TEST DATA
1473	006634	012203		MOV	(R2)+, R3	; GET DATA
1474	006636	020003		CMP	R0, R3	; CHECK
1475	006640	001403		BEQ	+10	
1476	006642	005046		CLR	-(SP)	; PUSH FAKE STATUS ON THE STACK
1477	006644	004767	172334	JSR	PC, ERROR	
1478	006650	005100		COM	R0	; COMPLEMENT CHECK WORD
1479	006652	005162	177776	COM	-2(R2)	; RESTORE DATA
1480	006656	005305		DEC	R5	; DECREMENT 4 WORD COUNTER
1481	006660	001350		BNE	6\$	
1482	006662	005100		COM	R0	; COMPLEMENT CHECK WORD
1483	006664	005304		DEC	R4	; DECREMENT 256. WORD COUNTER
1484	006666	001343		BNE	5\$	
1485	006670	005100		COM	R0	; COMPLEMENT CHECK WORD
1486	006672	005367	172070	DEC	COUNT	; DECREMENT BLOCK COUNTER
1487	006676	001335		BNE	4\$	
1488						
1489	006700	016602	000004	MOV	4(SP), R2	; GET BANK #
1490	006704	004767	177004	JSR	PC, STMM2	
1491	006710	016603	000002	MOV	2(SP), R3	; GET BLOCK COUNT
1492	006714	032737	040000	177570 BIT	#40000, 2#SWR	; LOOP ON TEST
1493	006722	001307		BNE	40\$; BRANCH IF LOOP ON TEST
1494	006724	006367	172024	ASL	ICOUNT	
1495	006730	102402		BVS	7\$	
1496	006732	000167	177354	JMP	1\$	
1497	006736	012705	000020	7\$: MOV	#16, R5	; COMPLEMENT PATTERN
1498	006742	011200		10\$: MOV	(R2), R0	; GET 1ST DATA WORD
1499	006744	016204	000020	MOV	20(R2), R4	; GET 9TH DATA WORD
1500	006750	110422		MOVB	R4, (R2)+	; SWAP WORDS 1-8
1501	006752	110422		MOVB	R4, (R2)+	; WITH 9-16
1502	006754	110422		MOVB	R4, (R2)+	
1503	006756	110422		MOVB	R4, (R2)+	
1504	006760	110422		MOVB	R4, (R2)+	
1505	006762	110422		MOVB	R4, (R2)+	
1506	006764	110422		MOVB	R4, (R2)+	
1507	006766	110422		MOVB	R4, (R2)+	
1508	006770	110422		MOVB	R4, (R2)+	
1509	006772	110422		MOVB	R4, (R2)+	
1510	006774	110422		MOVB	R4, (R2)+	
1511	006776	110422		MOVB	R4, (R2)+	
1512	007000	110422		MOVB	R4, (R2)+	

```

1513 007002 110422      MOVB    R4,(R2)+
1514 007004 110422      MOVB    R4,(R2)+
1515 007006 110422      MOVB    R4,(R2)+
1516 007010 110022      MOVB    R0,(R2)+      ;AND VICE VERSA
1517 007012 110022      MOVB    R0,(R2)+
1518 007014 110022      MOVB    R0,(R2)+
1519 007016 110022      MOVB    R0,(R2)+
1520 007020 110022      MOVB    R0,(R2)+
1521 007022 110022      MOVB    R0,(R2)+
1522 007024 110022      MOVB    R0,(R2)+
1523 007026 110022      MOVB    R0,(R2)+
1524 007030 110022      MOVB    R0,(R2)+
1525 007032 110022      MOVB    R0,(R2)+
1526 007034 110022      MOVB    R0,(R2)+
1527 007036 110022      MOVB    R0,(R2)+
1528 007040 110022      MOVB    R0,(R2)+
1529 007042 110022      MOVB    R0,(R2)+
1530 007044 110022      MOVB    R0,(R2)+
1531 007046 110022      MOVB    R0,(R2)+
1532 007050 005305      DEC     R5
1533 007052 001333      BNE    10$
1534 007054 005303      DEC     R3
1535 007056 001327      BNE    7$
1536
1537 007060 005767 171670      TST    ICOUNT
1538 007064 001402          BEQ    11$
1539 007066 000167 177220      JMP    1$
1540 007072 012616      11$:  MOV    (SP)+,(SP)
1541 007074 012616      MOV    (SP)+,(SP)
1542 007076 000207      RTS    PC
1543
1544      ;ROUTINE TO WRITE 8 XOR 13 WORST CASE NOISE TEST PATTERN
1545      .SBTTL 8 XOR 13 ROUTINES
1546      ;CALL: MOV    BANK#,-(SP)
1547      ;      MOV    #4KBANKS,-(SP)
1548      ;      JSR    PC,.8X13
1549
1550 007100 012616      .8X13: MOV    (SP)+,(SP)      ;ADJUST STACK
1551 007102 012616      MOV    (SP)+,(SP)
1552 007104 000207      RTS    PC
1553
1554      ;ROUTINE TO CHECK 8 XOR 13 WORST CASE NOISE TEST PATTERN
1555      ;CALL:
1556      ;      MOV    BANK#,-(SP)      ;PUSH FIRST BANK # ON THE STACK
1557      ;      MOV    #BANKS,-(SP)    ;PUSH # OF 4K BANKS TO CHECK ON THE STACK
1558      ;      JSR    PC,..8X13      ;CALL ROUTINE
1559
1560 007106 012616      ..8X13: MOV    (SP)+,(SP)
1561 007110 012616      MOV    (SP)+,(SP)
1562 007112 000207      RTS    PC      ;RETURN
1563
1564      .SBTTL ROTATING 1'S & 0'S ROUTINES
1565      ;ROUTINE TO CHECK ROTATING '0' BIT THROUGH FILED OF 1'S
1566      ;CALL: MOV    BANK#,-(SP)      ;SET STARTING BANK #
1567      ;      MOV    BLKCNT,-(SP)    ;SET 256. WORD BLOCK COUNT
1568      ;      JSR    PC,.R0T0      ;CALL ROUTINE

```

```

1569
1570 007114 004767 000276      .ROTO: JSR      PC,CKSWR      ;GO CHECK SWITCHES
1571 007120 016604 000002      MOV      2(SP),R4      ;GET 256. WORD BLOCK COUNT
1572 007124 016602 000004      MOV      4(SP),R2      ;GET FIRST BANK #
1573 007130 004767 176560      JSR      PC,STMM2      ;GO SET UP MEM MGMT (IF AVAIL)
1574 007134 012700 177777      MOV      #-1,R0        ;SET CHECK WORD
1575
1576 007140 012705 000400      1$:     MOV      #256.,R5      ;SET 256. WORD COUNT
1577 007144 000241 2$:     CLC                      ;CLEAR CARRY BIT IN PSW
1578 007146 004767 000124      JSR      PC,ROTATE      ;
1579 007152 016203 177776      MOV      -2(R2),R3      ;GET RESULT
1580 007156 103402 3$:     BCS      3$              ;BRANCH IF 'C' BIT WAS SET
1581 007160 020003 4$:     CMP      R0,R3          ;CHECK RESULT
1582 007162 001433 3$:     BEQ      4$              ;
1583 007164 005046 3$:     CLR      -(SP)          ;ERROR! COULD NOT ROTATE '0' BIT
1584 007166 004767 172012      JSR      PC,ERROR      ;THROUGH ADDRESS IN R2
1585 007172 005305 4$:     DEC      R5              ;DECREMENT 256. WORD COUNT
1586 007174 001363 2$:     BNE      2$              ;LOOP UNTIL DONE
1587 007176 005304 4$:     DEC      R4              ;DECREMENT 256. WORD BLOCK COUNT
1588 007200 001357 1$:     BNE      1$              ;LOOP UNTIL DONE
1589 007202 012616  (SP)+,(SP) ;POP CONSTANTS OFF THE STACK
1590 007204 012616  (SP)+,(SP) ;
1591 007206 000207      RTS      PC              ;RETURN TO CALLER
1592
1593      ;ROUTINE TO CHECK ROTATING '1' BIT THROUGH A FIELD OF 0'S
1594      ;CALL: MOV      BANK#,-(SP) ;SET STARTING BANK #
1595      ;      MOV      BLKCNT,-(SP) ;SET # OF 256. WORD BLOCKS TO CHECK
1596      ;      JSR      PC,.ROT1    ;CALL ROUTINE
1597
1598 007210 004767 000202      .ROT1: JSR      PC,CKSWR      ;GO CHECK SWITCHES
1599 007214 016604 000002      MOV      2(SP),R4      ;GET # OF 256. WORD BLOCKS TO CHECK
1600 007220 016602 000004      MOV      4(SP),R2      ;GET STARTING BANK #
1601 007224 004767 176464      JSR      PC,STMM2      ;GO SET UP MEM MGMT (IF AVAIL)
1602 007230 005000  CLR      R0            ;SET CHECK WORD
1603
1604 007232 012705 000400      1$:     MOV      #256.,R5      ;SET 256. WORD COUNTER
1605 007236 000261 2$:     SEC                      ;SET 'C' BIT IN PSW
1606 007240 004767 000032      JSR      PC,ROTATE      ;GO ROTATE '1' BIT
1607 007244 016203 177776      MOV      -2(R2),R3      ;GET RESULT
1608 007250 103002 3$:     BCC      3$              ;BRANCH IF 'C' IS CLEAR
1609 007252 020003 4$:     CMP      R0,R3          ;CHECK RESULT
1610 007254 001401 3$:     BEQ      .+4            ;
1611 007256 104400 3$:     HLT                      ;ERROR! COULD NOT ROTATE '1' BIT
1612      ;THROUGH ADDRESS IN R2
1613 007260 005305 4$:     DEC      R5              ;DECREMENT 256. WORD COUNT
1614 007262 001365 2$:     BNE      2$              ;
1615 007264 005304 4$:     DEC      R4              ;DECREMENT 256. WORD BLOCK COUNT
1616 007266 001361 1$:     BNE      1$              ;
1617 007270 012616  (SP)+,(SP) ;ADJUST RETURN ADDRESS
1618 007272 012616  (SP)+,(SP) ;
1619 007274 000207      RTS      PC              ;RETURN TO CALLER
1620
1621      ;ROUTINE TO ROTATE 'C' BIT THROUGH A MEMORY LOCATION.
1622 007276 106112 ROTATE: ROLB      (R2)      ;(R2)=177776 OR 000001
1623 007300 106112      ROLB      (R2)      ;(R2)=177775 OR 000002
1624 007302 106112      ROLB      (R2)      ;(R2)=177773 OR 000004

```



```

1625 007304 106112          ROLB   (R2)          ;(R2)=177767 OR 000010
1626 007306 106112          ROLB   (R2)          ;(R2)=177757 OR 000020
1627 007310 106112          ROLB   (R2)          ;(R2)=177737 OR 000040
1628 007312 106112          ROLB   (R2)          ;(R2)=177677 OR 000100
1629 007314 106112          ROLB   (R2)          ;(R2)=177777 OR 000000
1630 007316 106122          ROLB   (R2)+        ;(R2)=177577 OR 000200
1631 007320 106112          ROLB   (R2)          ;(R2)=177377 OR 000400
1632 007322 106112          ROLB   (R2)          ;(R2)=176777 OR 001000
1633 007324 106112          ROLB   (R2)          ;(R2)=175777 OR 002000
1634 007326 106112          ROLB   (R2)          ;(R2)=173777 OR 004000
1635 007330 106112          ROLB   (R2)          ;(R2)=167777 OR 010000
1636 007332 106112          ROLB   (R2)          ;(R2)=157777 OR 020000
1637 007334 106112          ROLB   (R2)          ;(R2)=137777 OR 040000
1638 007336 106112          ROLB   (R2)          ;(R2)=077777 OR 100000
1639 007340 106122          ROLB   (R2)+        ;(R2)=177777 OR 000000
1640 007342 000207          RTS     PC           ;RETURN
1641
1642
1643
1644
1645
1646
1647 007344 016604 000002          ;ROUTINE TO WRITE ONE WORD PATTERN INTO MEMORY
1648 007350 016602 000004          ;CALL: MOV   BANK# -(SP) ;PUSH STARTING BANK # ONTO STACK
1649 007354 004767 176334          ;      MOV   BLKCNT -(SP) ;AND 128. WORD BLOCK COUNT
1650 007360 012700          ;      JSR   PC,WRTPAT   ;CALL ROUTINE
1651 007362 000000          WRTPAT: MOV   2(SP),R4   ;GET BLOCK COUNT
1652 007364 012703 000100          ;      MOV   4(SP),R2   ;GET STARTING BANK #
1653 007370 010022          ;      JSR   PC,STMM2   ;GO SET UP MEM MGMT
1654 007372 010022          ;      MOV   (PC)+,R0   ;GET USER CONSTANT
1655 007374 010022          .CONST: 0
1656 007376 010022          1$: MOV   #64,R3       ;SET 256. WORD COUNTER
1657 007400 005303          2$: MOV   R0,(R2)+    ;WRITE 256. WORDS
1658 007402 001372          ;      MOV   R0,(R2)+
1659 007404 005304          ;      MOV   R0,(R2)+
1660 007406 001366          ;      MOV   R0,(R2)+
1661 007410 012616          ;      DEC   R3         ;DECREMENT 256. WORD COUNTER
1662 007412 012616          ;      BNE   2$         ;LOOP UNTIL 256. WORDS HAVE BEEN WRITTEN
1663 007414 000207          ;      DEC   R4         ;DECREMENT BLOCK COUNT
1664
1665
1666
1667
1668
1669 007416 042767 017777 171334          ;ROUTINE TO CHECK THE SWITCH REGISTER
1670 007424 032737 000400 177570          ;CHECK SWITCH 9: IF SET, LOAD ERROR COUNT INTO THE DISPLAY REGISTER;
1671 007432 001402          ;IF NOT SET, LOAD PASS COUNT INTO THE DISPLAY REGISTER
1672 007434 004767 000464          CKSWR: BIC   #17777,LDISP ;SAVE RELOCATION BITS
1673 007440 032737 001000 177570          ;      BIT   #BIT8,2$SWR ;CHECK SWITCH 8
1674 007446 001404          ;      BEQ  10$        ;BRANCH IF SET
1675 007450 056767 171302 171302          ;      JSR   PC,REL24K  ;GO RELOCATE PROGRAM BACK TO 4K AND STOP
1676 007456 000403          10$: BIT   #BIT9,2$SWR ;SWITCH 9 SET ?
1677 007460 056767 171266 171272          ;      BIS   ERCNT,LDISP ;LOAD ERROR COUNT
1678 007466 016737 171266 177570          ;      BR   2$         ;
1679 007474 012767 040177 171252          1$: BIS   ICNT,LDISP  ;LOAD PASS COUNT
1680 007502 032737 004000 177570          2$: MOV   LDISP,2$DISPLAY ;LOAD THE DISPLAY REGISTER
                                ;      MOV   #040177,ICOUNT ;LOAD ITERATION COUNT WORD
                                ;      BIT   #4000,2$SWR   ;CHECK SW11
    
```

1681	007510	001402		
1682	007512	105067	171236	
1683	007516	000207		
1684				
1685				
1686	007520	005015	047524	051040
1687	007526	051505	047524	042522
1688	007534	046040	040517	042504
1689	007542	051522	051440	040524
1690	007550	052122	040440	020124
1691	007556	033061	006462	000012
1692	007564	005015	047105	041101
1693	007572	042514	050040	051101
1694	007600	052111	037531	030440
1695	007606	030057	054475	051505
1696	007614	047057	020117	000
1697	007621	015	051412	040524
1698	007626	052122	047111	020107
1699	007634	040502	045516	021440
1700	007642	034050	037451	000040
1701	007650	005015	020043	043117
1702	007656	032040	020113	040502
1703	007664	045516	020123	047524
1704	007672	052040	051505	024124
1705	007700	024470	020077	000
1706	007705	015	050012	052101
1707	007712	042524	047122	021440
1708	007720	020077	000	
1709	007723	015	037412	000
1710	007727	015	052012	050131
1711	007734	020105	047503	051516
1712	007742	040524	052116	000
1713	007747	015	044412	050116
1714	007754	052125	021440	047440
1715	007762	020106	032462	027066
1716	007770	053440	051117	020104
1717	007776	046102	041517	051513
1718	010004	052040	020117	042524
1719	010012	052123	044440	051516
1720	010020	042524	042101	047440
1721	010026	000106		
1722	010030	005015	054524	042520
1723	010036	040440	042104	042522
1724	010044	051523	000	
1725	010047	015	052012	020117
1726	010054	042522	052123	051117
1727	010062	020105	051120	043517
1728	010070	040522	020115	052123
1729	010076	051101	020124	052101
1730	010104	000040		
1731	010106	000052		
1732	010110	042504	045115	020101
1733	010116	047504	042516	000041
1734				
1735				
1736				

```

BEQ      .+6
CLR      ICOUNT
RTS      PC
;ICOUNT =040000 IF SW11 =1

:MESSAGES
RESLDR: .ASCIZ <15><12>'TO RESTORE LOADERS START AT 162'<15><12>

PARITY: .ASCIZ <15><12>'ENABLE PARITY? 1/0=YES/NO '

STBANK: .ASCIZ <15><12>'STARTING BANK #(8)? '

BANKS:  .ASCIZ <15><12>'# OF 4K BANKS TO TEST(8)? '

PAT:    .ASCIZ <15><12>'PATTERN #' '

QUEST:  .ASCIZ <15><12>'?'
CONST:  .ASCIZ <15><12>'TYPE CONSTANT'

PRG3M:  .ASCIZ <15><12>'INPUT # OF 256. WORD BLOCKS TO TEST INSTEAD OF'

PRG4M:  .ASCIZ <15><12>'TYPE ADDRESS'

RELOCM: .ASCIZ <15><12>'TO RESTORE PROGRAM START AT '

ASTERISK: .ASCIZ '*'
ENDMSG:  .ASCIZ 'DEMJA DONE!'

.EVEN

;ROUTINE TO RELOCATE PROGRAM BACK TO 0

```

1737	010124	010700		REL24K:	MOV	PC,RO		;FORM BASE ADDRESS WHERE CODE
1738	010126	042700	017777		BIC	#17777,RO		;IS RELOCATED
1739	010132	010067	000004		MOV	RO,1\$;PUT FROM ADDRESS INTO SUBROUTINE CALL
1740	010136	004567	174132		JSR	RS,RELOC		;RELOCATE CODE TO
1741	010142	000000		1\$:	0			;LOWEST 4K
1742	010144	000000			0			
1743	010146	012706	000500		MOV	#STKPTR,SP		;SET STACK PTR
1744	010152	042737	100000	000760	BIC	#100000,@#LDDISP		;CLEAR RELOCATION INDICATOR
1745	010160	013737	000760	177570	MOV	@#LDDISP,@#DISPLAY		;LOAD DISPLAY REGISTER
1746	010166	005037	000764		CLR	@#RELOCF		;CLEAR RELOCATION FACTOR
1747	010172	000005			RESET			;DISABLE MEM MGMT
1748	010174	000137	000162		JMP	@#PONE		;RESTORE LOADERS & HALT
1749								
1750	010200			LODAR:				
1751		000001			.END			

E04

MAINDEC-11-DEMJA-C PDP11/70 MEMORY TEST MACY11 30(1046) 12-JUL-77 10:09 PAGE 38
DEMJA.P11 16-MAY-77 13:58 CROSS REFERENCE TABLE -- USER SYMBOLS

.3151	006224	1346#		
.3159	006252	1348	1350	1354#
.3N0T9	006242	1347	1351#	
.3X9	006064	821	1297#	
.8X13	007100	851	1550#	

F04

MAINDEC-11-DEMJA-C PDP11/70 MEMORY TEST MACY11 30(1046) 12-JUL-77 10:09 PAGE 40
DEMJAC.P11 16-MAY-77 13:58 CROSS REFERENCE TABLE -- MACRO NAMES

\$TYPE 48

. ABS. 010200 000

ERRORS DETECTED: 0

DSKZ:DEMJAC.BIN DSKZ:DEMJAC.LST/CRF/SOL/NL:TOC=DSKZ:DEMJAC.P11

RUN-TIME: 43.4 SECONDS

RUN-TIME RATIO: 154/8=18.2

CORE USED: 25K (49 PAGES)