

MF11S-K

MF11S-K MEM DIAG
CZMMLCO

AH-0145C-MC
FICHE 1 OF 1

MAY 1980
COPYRIGHT © 77 79
MADE IN USA



A grid of 12 columns and 12 rows of small, illegible text blocks, likely representing a memory diagnostic or data table. The text is too small to be transcribed accurately.



IDENTIFICATION

PRODUCT CODE: AC-0144C-MC
PRODUCT NAME: CZMMLCO MF11S-K MEM DIAG
PRODUCT DATE: OCTOBER 1979
MAINTAINER: DIAGNOSTIC ENGINEERING
AUTHOR: JAMES P. RYAN

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

THE SOFTWARE DESCRIBED IN THIS DOCUMENT IS FURNISHED 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) 1977,1979 BY DIGITAL EQUIPMENT CORPORATION

CONTENTS

1.0	ABSTRACT
1.1	GETTING STARTED
2.0	REQUIREMENTS
2.1	EQUIPMENT
2.2	STORAGE
3.0	LOADING PROCEDURE
4.0	STARTING PROCEDURE
4.1	SWITCH SETTINGS
4.2	CONTROL-C FUNCTION
4.3	STARTING ADDRESS =200 RESTART ADDRESS =250
4.4	PROGRAM AND/OR OPERATOR ACTION
5.0	PROGRAM HALTS (NORMAL + ERROR)
6.0	ERRORS
6.1	ERROR MESSAGE FORMAT.
6.2	ERROR DICTIONARY
6.3	ERROR HISTORY
6.4	ERROR RECOVERY
6.5	MOS CHIP CROSS REFERENCE TABLE
7.0	RESTRICTIONS
8.0	MISCELLANEOUS
8.1	ADDRESS/BANK RANGES IN OCTAL AND DECIMAL
8.2	EXECUTION TIME
8.3	PASS COUNT AND TEST NO. LOCATIONS
8.4	STACK POINTER
8.6	POWER FAIL
9.0	PROGRAM DESCRIPTION
9.1	NARRATIVE FLOW CHART
9.2	TEST TITLES AND TEST TIMES
10.0	RXDP & ACT11 & APT OPERATION
11.0	ECO HISTORY

THIS DIAGNOSTIC WILL TEST 0 - 124K OF MF11S-K MEMORY ON ANY PDP-11 FAMILY COMPUTER THAT DOES NOT PERFORM A DATIP CYCLE BEFORE A DATO CYCLE FOR ALL INSTRUCTIONS.

THIS PROGRAM CAN BE RUN UNDER XXDP, APT AND ACT MONITORS. ON PROCESSORS WITH NO HARDWARE SWITCH REGISTER, SOFTWARE SWITCH REGISTER = LOCATION 176 AND SOFTWARE DISPLAY REGISTER = LOCATION 174.

[1.1] GETTING STARTED

IF NO HARDWARE SWITCH REGISTER SET LOCATION 176 TO OBTAIN SWITCH OPTIONS.
TO START:

-
- A. SET SWITCH REGISTER = 00000
 - B. START AT 200.
 - C. THE ADDRESS OF THE CONTROL AND STATUS REGISTER (CSR) AND THE MEMORY LIMITS IT CONTROLS WILL BE PRINTED. IF TWO CONTROLLERS ARE PRESENT, BOTH CSR ADDRESSES WILL BE TYPED AS WELL AS BOTH MEMORY LIMITS.
 - D. SEE SECTION 4.4 FOR REST OF PRINTOUTS EXPECTED.
 - E. 'END PASS #01' WILL BE TYPED. THIS IS A QUICK VERIFY PASS AND THE TEST IS NOT COMPLETE UNTIL 'END PASS #02' IS TYPED.
 - F. TO HALT THE TEST, TYPE CONTROL-C, THIS WILL INSURE THE PROGRAM IS RELOCATED BACK TO LOWER MEMORY AND THAT THE MEMORY HAS BEEN PURGED OF ANY DOUBLE ERRORS FORCED DURING TESTING. BE PATIENT, THE CONTROL-C IS ONLY RECOGNIZED AT THE END OF THE CURRENT SUBTEST.
 - G. IF AN UNEXPECTED HALT OCCURS SEE SECTION 6.0. IF AN ERROR # IS TYPED SEE SECTION 6.2.

!CAUTION! BEFORE 'DIGGING' INTO THE LISTING READ SECTION 9.

SWITCH SETTING SUMMARY (SEE SECTION 4.1 FOR DETAILS)

BIT15(100000)	HALT ON ERROR
BIT14(040000)	LOOP IN SUBTEST DEFINED BY BITS <4:0>
BIT13(020000)	INHIBIT ERROR PRINTOUTS
BIT12(010000)	DISABLE TESTING ABOVE 28K (MEMORY MANAGEMENT)
BIT11(004000)	ENABLE PRINTOUTS OF SINGLE ARRAY ERRORS
BIT10(002000)	HALT AFTER EACH SUBTEST
BIT09(001000)	INHIBIT PROGRAM RELOCATION
BIT08(000400)	TYPE FIRST FAILING BIT ERROR PER 4K.
BIT07(000200)	ENABLE XOR PRINTOUT FOR ARRAY TESTS
BIT06(000100)	INHIBIT MEMORY SIZING
BIT05(000040)	INHIBIT 'END PASS #XX' AND 'RELOC' PRINTOUTS
BIT04	TO CLEAR MEMORY OF DBE'S(NO TESTS RUN)
BIT03-BIT00	BEGINNING TEST NUMBER.

[2.0] REQUIREMENTS

[2.1] EQUIPMENT

STANDARD 11 FAMILY COMPUTER WITH A CONSOLE OUTPUT DEVICE
AND FROM 32K TO 124K OF MS11K MEMORY (WITH EXCEPTION IN 1.0) .

[2.2] STORAGE

PROGRAM STORAGE - 0000 - 14200. PROGRAM EXPANDS FOR ERROR
HISTORY AND TO SAVE ABSOLUTE LOADER OR XXDP CHAIN MONITOR.
(SEE SECTION 9. FOR DETAILS)

[3.0] LOADING PROCEDURE

USE STANDARD PROCEDURE FOR PDP-11 ABSOLUTE BINARY FORMATTED TAPES.
STARTING PROCEDURE

[4.1] SWITCH SETTINGS

SOFTWARE SWITCH REGISTER = LOCATION 176

BIT15(100000) HALT ON ERROR (SEE 5.0)

BIT14(040000) LOOP ON TEST DEFINED BY SWITCH REGISTER BITS <4:0>
BIT13(020000) INHIBIT ERROR PRINTOUTS AND ERROR HISTORY PRINTOUTS.
ERROR HISTORY CAN STILL BE OBTAINED BY TYPING ^C.
BIT12(010000) DISABLE TESTING ABOVE 28K. (MEMORY MANAGEMENT)

BIT11(004000) ENABLE PRINTOUTS OF SINGLE ERRORS ON ARRAY TESTS
(DISABLES ECC FOR ARRAY TESTS)

BIT10(002000) HALT AFTER EACH SUBTEST
!PRESS CONTINUE TO DO NEXT SUBTEST
BIT09(001000) INHIBIT PROGRAM RELOCATION
!IF SET LOCATIONS 430-7776 WILL NOT BE
!TESTED.

BIT08(000400) TYPE FIRST UNCORRECTABLE ERROR IN EACH 4K BANK ONLY.
!THE TOTAL ERROR COUNT (UP TO 377) WILL
!BE SAVED IN THE ERROR HISTORY.

BIT07(000200) ENABLE XOR PRINTOUT FOR ARRAY TESTS
BIT06(000100) INHIBIT MEMORY SIZING.
!THE MEMORY LIMITS MUST BE SETUP IN THE FOLLOWING LOCATIONS:
(VALUES TO TEST 0-28K ARE SHOWN)
(LOWTWO=LOCATION 332)

LOWTWO: 0 ;STORE BITS 17:16 OF LOW TEST ADDRESS
LOWADD: 0 ;STORE REST OF LOW TEST ADDRESS
HIGHTWO: 0 ;STORE BITS 17:16 OF HIGH TEST ADDRESS
HIGHADD: 157776 ;STORE REST OF HIGH TEST ADDRESS

NOTE: HIGHADD MUST BE SET TO A 4K BOUNDARY. (E.G. 37776)
AND LOCATION LDFLG MUST = 1 IF THE PROGRAM
IS RESIDING IN THE MS11K MEMORY.

BIT05(000040) INHIBIT 'END PASS #XX' AND 'RELOC' PRINTOUTS F 1
BIT04 SCANS MEMORY WITH ECCDIS. NO TESTS RUN.
TERMINATE AFTER PASS#1 WITH ^C.
BIT03-BIT00 NUMBER OF TEST (0-17) TO RUN FIRST.
!NORMALLY USED WITH BIT14 (LOOP ON TEST)

SEQ 0005

[4.2] CONTROL-C FUNCTION

CONTROL C [^C] AFTER COMPLETION OF THE CURRENT TEST.
THE ERROR HISTORY (SEE SEC. 6.3) WILL BE
TYPED. THE PROGRAM WILL LOOP IN LOWER MEMORY.
HALT AND RESTART AT 250 IF DESIRED.

IN ORDER TO COMPLETELY TEST THE ERROR CORRECTING
LOGIC, DOUBLE ERRORS ARE FORCED INTO MEMORY ON
CERTAIN TESTS. TO EXIT FROM THE PROGRAM AND HALT
EXECUTION, ^C MUST BE TYPED AND THE TEST
PERMITTED TO CLEAN UP THE MEMORY UNDER TEST. IF
NOT, THERE EXISTS THE POSSIBILITY OF UNCORRECT-
ABLE ERRORS REMAINING IN THE MEMORY.

[4.3] STARTING ADDRESS= 200
RESTART ADDRESS = 250 OR 200

RESTART AT 200 CLEARS PASS COUNT (\$PASS) AND PRINTS DIAGNOSTIC TITLE.

[4.4] PROGRAM AND/OR OPERATOR ACTION

- 1) LOAD PROGRAM INTO MEMORY USING ABSOLUTE LOADER.
- 2) SET OPTIONS (SEE SEC. 4.1)
- 3) START THE PROGRAM AT 200
- 4) THE FOLLOWING IS AN EXAMPLE WITH EXPLANATIONS
OF THE PRINTOUTS EXPECTED.

'XXXXX-YYYYY' ;ADDRESSES OF TEST BOUNDARIES.

'NO RELOC-SBE IN LOC 0-500' ;RELOCATION IS INHIBITED BECAUSE
;SINGLE ERRORS IN THIS AREA WILL CORRUPT
;PROGRAM WHEN RELOCATED.

'RELOC' ;THE DIAGNOSTIC RELOCATES TO HIGHEST
;BANK UNDER TEST. AND RUNS TST0-TST13 AGAIN.

'END PASS #XX' ;WHERE 'XX' IS THE PASS NO.

'TST7-NO ERROR FREE LOC IN SLICE AT XXXXXX' THE LAST PART OF
TEST 7 CHECKS FOR PROPER 1K ADDRESSES IN
THE CSR AFTER A DOUBLE ERROR. THIS MES-
SAGE INDICATES THAT THIS SLICE COULD NOT
BE TESTED AND IT IS PROCEEDING TO THE
NEXT 1K SLICE.

'NO MNG' :PRINTED IF TESTING ABOVE 28K IS ENABLED
 : (BIT 12 OF SWR) AND MEMORY MANAGEMENT IS
 : NOT AVAILABLE.

[5.0] PROGRAM HALTS (NORMAL+ ERROR)

THIS IS A LIST OF EXPECTED HALTS. IF THE TEST HALTS
 IN A LOCATION NOT IN THIS LIST AND ITS LESS THAN 776, IT
 MAY BE DUE TO A DEVICE INTERRUPTING.

NOTE THE HALT AT END OF SUBTEST AND HALT ON ERROR HALT LOCATIONS
 MAY BE RELOCATED. THE ACTUAL LOCATIONS THEY ARE IN CAN BE FOUND
 BY SUBTRACTING 500 FROM 2362[SWHALT] AND ADDING THIS DIFFERENCE TO THE
 CONTENTS OF SAVR6 [LOC. 352].

PC --	REASON -----	RECOVERY -----
112	TRAP TO LOC. 4	EXAMINE R6, IT CONTAINS THE POINTER TO THE PC WHERE THE TRAP OCCURRED. (SEE NOTE) THE PROGRAM WILL LOOP AT LOC BUSER UNTIL MANUALLY HALTED. CHECK THE LOCATION POINTED TO BY MINMEM(324) TO INSURE THAT NO DBE'S REMAIN IN THE TESTED AREA.
100	POWER FAIL	POWER UP WILL RECOVER IF IN CORE MEMORY.
2364	HALT AT END OF TEST SWITCH SET.	PRESS CONTINUE TO GO TO NEXT SUBTEST.
13624	HALT ON ERROR	RESTART AT 250
13712	CONTROL-C TYPED OR FATAL ERROR OCCURRED	RESTART AT 250

NOTE: ANY OF THE ABOVE HALT CONDITIONS THAT PERTAIN TO
 APT ACTUALLY RESULT IN A TIGHT LOOP SO THAT THE
 ERROR INFORMATION CAN BE RECOVERED. CONSULT LISTING
 FOR MORE DETAIL, IF NECESSARY.

[6.0] ERRORS

ANY REFERENCE TO 'ERROR' IN THIS DOCUMENT INDICATES AN ERROR
 AS DEFINED BY THE PROGRAM AT EXECUTION TIME. NORMALLY, IT IS
 AN UNCORRECTABLE ERROR UNLESS ERROR CORRECTING IS DISABLED
 THEN IT MEANS ALL ERRORS.

[6.1] ERROR MESSAGE FORMAT

H 1

SEQ 0007

THE ERROR PRINTOUT CONSISTS OF 6 OCTAL WORDS IN THE FOLLOWING
FORMAT:

"ADDR GOOD BAD PC ERR # PASFLG XOR "

"ADR ERR" WILL BE PRINTED PRIOR IF AN ADDRESSING ERROR IS SUSPECTED.

WHERE:

ADDR = FAILING MEMORY LOCATION
GOOD = GOOD DATA [DATA THAT WAS EXPECTED]
BAD = BAD DATA [DATA THAT WAS FOUND]
PC = PROGRAM COUNTER AT ERROR CALL.
ERR # = FAILING ERROR NO. (SEE SEC 6.2 - ERROR DICTIONARY)
PASFLG = CONTENTS OF LOCATION PASFLG. THIS MAY NOT BE RELEVANT.
(SEE SEC. 6.2-ERROR DICTIONARY)
XOR = EXCLUSIVE 'ORING' OF GOOD VS. BAD (BITS IN ERROR).
AVAILABLE ONLY IF SINGLE ERRORS ARE TO BE PRINTED
OUT VIA SWREG BIT 07 HIGH.

!THE TEST WILL CONTINUE AFTER THE ERROR PRINTOUT.
!"NO MNG" WILL BE TYPED IF TESTING ABOVE 28K SELECTED AND NO MEMORY
!MANAGEMENT IS FOUND.

(FATAL ERRORS)

"ERROR #XXXXXX" WILL BE TYPED WHERE "XXXXXX" IS
THE ERROR NUMBER. THE DIAGNOSTIC WILL LOOP AT FATHLT UNTIL MANUALLY
HALTED. SEE SEC. 6.2 -ERROR DICTIONARY - FOR DESCRIPTIONS
OF THE ERROR.

(APT MODE ERRORS)

ALL ERRORS ARE TREATED AS FATAL UNDER APT. WHEN AN
ERROR OCCURS UNDER APT A "1" IS STORED IN LOCATION
\$MSGTY AND THE PROGRAM LOOPS AT FATHLT.

\$FATAL CONTAINS THE ERROR NO. IN THE LOW BYTE AND
THE FAILING BANK NO. UNDER TEST IN THE HIGH BYTE.

THIS IS A LIST OF ERROR NUMBERS PRINTED AND POSSIBLE CAUSES FOR THE ERROR. IF THE ERROR IS SUCH THAT THE TESTING CAN BE CONTINUED AFTER THE ERROR PRINTOUT, IT WILL DO SO.

THE ROUTINE NAME WHERE THE ERROR CALL ORIGINATED IS GIVEN IN BRACKETS.

NOTE- 'BAKPAT' REFERS TO THE BACKGROUND PATTERN WRITTEN INTO MEMORY FOR VARIOUS TESTS. BAKPAT HAS THE VALUE = 377, AND SWAPAT HAS THE VALUE = 177400.

@

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

```
.ABS  
.NLIST MD,MC,CND  
.LIST ME,BIN,SEQ,LOC  
.TITLE CZMMLCO MF11S-K MEM DIAG  
:*COPYRIGHT (C) FEB 1977,OCT 1979  
:*DIGITAL EQUIPMENT CORP.  
:*MAYNARD, MASS. 01754  
:*  
:*PROGRAM BY JIM RYAN  
:*  
:*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC  
:*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.  
:*  
$SWR=160000 ;:HALT ON ERROR, LOOP ON TEST, INHIBIT ERROR TYP0UT
```

;:TRAP CATCHER OF .+2 AND HALT FOR 0-776 LOCATIONS

000240
000042 000042 000000
000044 000046 000156 000052 040000 000044
000044 016777 002640 002656
000052 005077 000242
000056 005077 000240
000062 042701 000003
000066 005011
000070 005061 000002
000074 012737 000136 000024
000102 000000

```
SCOPE =NOP  
.=42  
.WORD 0 ;FOR ACT/XXDP  
.SBTTL ACT11 HOOKS  
;:*****  
;HOOKS REQUIRED BY ACT11  
$SVPC=. ;SAVE PC  
.=46  
$ENDAD ;:1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP  
.=52  
.WORD 40000 ;:2)SET LOC.52 TO 40000  
.= $SVPC ;: RESTORE PC  
PWRDN: MOV ECCDIS,@CSRADR ;DISABLE ECC  
CLR @MINMEM ;CLEAR FIRST ADDRESS  
CLR @MINMEM+2 ;SECOND WORD  
BIC #3,R1  
CLR (R1) ;CLR A LOC  
CLR 2(R1) ;NEXT  
MOV #PWRUP,@#24  
HALT
```



```
105 000214 005043      1$: CLR      -(R3)      ;CLEAR A MAILBOX LOCATION
106 000216 022703 000400  CMP      #$MAIL,R3    ;DONE?
107 000222 001374      BNE      1$           ;BRANCH IF NO
108 000224 105737 000042  TSTB    @#42         ;ACT11 MODE?
109 000230 001007      BNE      RESTRT      ;BRANCH IF YES
110 000232 105737 000170  TSTB    @#REL        ;ARE WE RELOCATED?
111 000236 100404      BMI      RESTRT      ;BR IF YES- SINCE TPCRLF IS RELOCATED ALSO-
112 000240 004767 015474  JSR     PC,PRITL     ;PRINT DIAGNOSTIC TITLE
113 000244 000240      NOP                    ;SO WE FALL THRU TO RESTART
114 000246 000240      NOP                    ;DITTO
115
116 000250 012703 000344  RESTRT: MOV     #SAVR5,R3 ;POINT R3 TO THE LOC SAVR5
117 000254 012305      MOV     (R3)+,R5     ;RESTORE R5
118 000256 012306      MOV     (R3)+,SP    ;AND RESTORE R6 JUST IN CASE IT IS A RESTART
119 000260 010600      MOV     SP,R0       ;PLACE THE STARTING ADDRESS OF THE TEST IN R0
120 000262 012746 000340  MOV     #340,-(SP)  ;SET HIGH PRIORITY FOR RTI
121 000266 010046      MOV     R0,-(SP)
122 000270 000002      RTI                    ;GO TO 'START'-MAY BE RELOCATED.
123                                     ;IF RELOCATED SEE LOCATION SAVR6 FOR START.
124
125
126
127
128
129
130
131
```

.SBTTL APT PARAMETER BLOCK

```
132                                     ;*****
133                                     ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
134                                     ;*****
135                                     ;$X=      ;;SAVE CURRENT LOCATION
136                                     ;=24      ;;SET POWER FAIL TO POINT TO START OF PROGRAM
137 000272                                     200      ;;FOR APT START UP
138 000024                                     ;=44      ;;POINT TO APT INDIRECT ADDRESS PNTR.
139 000200                                     $APTHDR  ;;POINT TO APT HEADER BLOCK
140 000044                                     ;=. $X    ;;RESET LOCATION COUNTER
141 000272                                     ;*****
142 000272                                     ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
143                                     ;INTERFACE SPEC.
144
145 $APTHD:
146 $HIBTS: .WORD 0      ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
147 $MBADR: .WORD $MAIL  ;;ADDRESS OF APT MAILBOX (BITS 0-15)
148 $STMT:  .WORD 25    ;;RUN TIM OF LONGEST TEST
149 $PASTM: .WORD 110.  ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
150 $UNITM: .WORD      ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
151 000304 .WORD $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
152
153
154
155
156 000272      MMAVA:  .=$APTHD      ;THIS BYTE IS USED TO DETERMINE IF MEMORY
157 000272                                     ;MANAGEMENT IS AVAILABLE OR NOT
158
159
160 000273      .=$MMAVA+1
```

161	000273	TYPENB:		;THIS BYTE IS USED TO DETERMINE IF THE
162				;TYPE OUT OF ERROR HAS BEEN ENABLED OR NOT
163				
164	000274	SPRERR: . =TYPENB+1		
165	000274			;THIS BYTE DETERMINES IF THE PROGRAM HAS FOUND
166				;A PARITY ERROR
167				
168	000275	\$ADERR: . =SPRERR+1		
169	000275			;THIS BYTE IS USED TO DETERMINE IF THE
170				;PROGRAM HAS ENCOUNTERED ADDRESS ERROR
171				
172	000276	STRTDI: . =\$ADERR+1		
173	000276			
174	000300	LOWBNK: . =STRTDI+2		
175	000300			
176	000302	PASFLG: . =LOWBNK+2		
177	000302			;LOWER BYTE OF THIS WORD GIVES THE PASS NUMBER FOR
178				;THE SPECIFIC TEST WHEREAS THE UPPER BYTE
179				;HAS BEEN USED BY DIFFERENT TEST FOR DIFFERENT PURPOSES
180				
181	000304	ENDSTK: . =PASFLG+2		
182	000304			
183	000306	PBNK: . =ENDSTK+2		
184	000306	DECWRD:		;HOLDS BANK UNDER TEST FOR 'TST BNK XX' PRINTOUT.
185	000306			
186	000310	TYPCNT: . =DECWRD+2		
187	000310	.BYTE 0		;THIS BYTE DETERMINES THE NUMBER OF WORDS
188				;TO BE TYPED
189	000311	SAVKBB: .BYTE 0		;THIS LOCATION IS USED TO SAVE THE CHARACTER
190				;HIT BY THE OPERATOR
191				;ALSO IS USED AS TEMP IN ROUTINE \$GTSIZ.
192		.EVEN		
193				
194				
195	177560	TKS= 177560		
196	177562	\$KBB= 177562		
197	177564	\$TPS= 177564		
198	177566	\$TPB= 177566		
199	177572	SRO= 177572		
200	000312	BAKPAT: .WORD 377		;BACKGROUND PATTERN WRITTEN TO MEMORY.
201				
202	000314	SWAPAT: .WORD		
203	000316	RELBOT: BEGIN-50		;HOLDS LOWEST TEST ADDRESS WHEN RELOCATED.
204	000320	MINMEM: .WORD 100000		;FIRST ADDR FOR ECC IN CASE OF APT
205				
206		::*****		
207		;LOCATIONS TO BE MODIFIED IF LIMITS SET BY OPERATOR		
208	000322	LOWTWO: 0		;HOLDS BITS 17:16 OF LOW TEST ADDRESS
209	000324	LOWADD: 0		;HOLDS BITS 15:0 OF LOW TEST ADDRESS
210				
211	000326	HIGHTWO: 0		;HOLDS BITS 17:16 OF HIGH TEST ADDRESS
212	000330	HIGHADD: 37776		;HOLDS BITS 15:0 OF HIGH TEST ADDRESS
213		::*****		
214				
215	000332	\$HIMAX: 0		;HOLDS BITS 17:16 OF MAXIMUM AVAILABLE MEMORY
216	000334	\$MAXM: 17776		;HOLDS BITS 15:0 OF MAXIMUM AVAILABLE MEMORY


```
217  
218 000336 000000 MAXMEM: .WORD ;MAXIMUM CURRENT VIRTUAL MEMORY UNDER TEST  
219  
220 000340 000000 SAVMAX: .WORD  
221 000342 000000 SAVR4: .WORD  
222 000344 000000 SAVR5: .WORD  
223  
224 ;* SAVR6 POINTS TO WHERE THE PROGRAM STARTS EVEN WHEN RELOCATED.  
225 000346 000500 SAVR6: .WORD BEGIN ;CONTAINS START ADDRESS WHEN RELOCATED ALSO.  
226 000350 000000 SAVLOC: 0 ;TEST 17 STORES ERROR INFO HERE  
227  
228 ;GET HERE IF AN ILLEGAL TRAP TO LOC. 4 OCCURRED  
229  
230 000352 012737 000001 000400 BUSER: MOV #1,@#MSGTY ;TELL APT FATAL ERROR #000  
231 000360 000774 BR BUSER ;LOOP SO APT CAN GET INFO  
232  
233  
234 ;*364-400 IS USED AS A STACK AREA BY ERRCHK ROUTINE FOR ERROR HISTORY PRINTOUT
```

235 000400
236
237
238
239
240 000400
241 000400 000000
242 000402 000000
243 000404 000000
244 000406 000000
245 000410 000000
246 000412 000000
247 000414 000000
248 000416 000000
249 000420
250 000420 000
251 000421 000
252 000422 000000
253 000424 000000
254 000426 000000
255
256
257
258
259
260
261 000430 000
262 000431 000
263
264
265
266
267 000432 000000
268
269 000434 000
270 000435 000
271 000436 000000
272 000440 000
273 000441 000
274 000442 000000
275 000444 000
276 000445 000
277 000446 000000
278 000450
279
280
281
282
283
284 000500 000500
285 000500 010706
286 000502 005746
287 000504 012704 016154
288 000510 010637 000346
289 000514 012737 000044 000024
290 000522 004767 014562

.=400
.SBTTL APT MAILBOX-ETABLE

::*****

.EVEN
\$MAIL: ;: APT MAILBOX
\$MSGTY: .WORD AMSGTY ;: MESSAGE TYPE CODE
\$FATAL: .WORD AFATAL ;: FATAL ERROR NUMBER
\$TESTN: .WORD ATESTN ;: TEST NUMBER
\$PASS: .WORD APASS ;: PASS COUNT
\$DEVCT: .WORD ADEVCT ;: DEVICE COUNT
\$UNIT: .WORD AUNIT ;: I/O UNIT NUMBER
\$MSGAD: .WORD AMSGAD ;: MESSAGE ADDRESS
\$MSGLG: .WORD AMSGLG ;: MESSAGE LENGTH
\$ETABLE: ;: APT ENVIRONMENT TABLE
\$ENV: .BYTE AENV ;: ENVIRONMENT BYTE
\$ENVM: .BYTE AENVM ;: ENVIRONMENT MODE BITS
\$SWREG: .WORD ASWREG ;: APT SWITCH REGISTER
\$USWR: .WORD AUSWR ;: USER SWITCHES
\$CPUOP: .WORD ACPUOP ;: CPU TYPE, OPTIONS
BITS 15-11=CPU TYPE
11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
11/70=06,PDQ=07,Q=10
BIT 10=REAL TIME CLOCK
BIT 9=FLOATING POINT PROCESSOR
BIT 8=MEMORY MANAGEMENT
\$MAMS1: .BYTE AMAMS1 ;: HIGH ADDRESS,M.S. BYTE
\$MTYP1: .BYTE AMTYP1 ;: MEM. TYPE,BLK#1
MEM.TYPE BYTE -- (HIGH BYTE)
900 NSEC CORE=001
300 NSEC BIPOLAR=002
500 NSEC MOS=003
\$MADR1: .WORD AMADR1 ;: HIGH ADDRESS,BLK#1
MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF 'TYPE' ABOVE
\$MAMS2: .BYTE AMAMS2 ;: HIGH ADDRESS,M.S. BYTE
\$MTYP2: .BYTE AMTYP2 ;: MEM. TYPE,BLK#2
\$MADR2: .WORD AMADR2 ;: MEM.LAST ADDRESS,BLK#2
\$MAMS3: .BYTE AMAMS3 ;: HIGH ADDRESS,M.S.BYTE
\$MTYP3: .BYTE AMTYP3 ;: MEM. TYPE,BLK#3
\$MADR3: .WORD AMADR3 ;: MEM.LAST ADDRESS,BLK#3
\$MAMS4: .BYTE AMAMS4 ;: HIGH ADDRESS,M.S.BYTE
\$MTYP4: .BYTE AMTYP4 ;: MEM. TYPE,BLK#4
\$MADR4: .WORD AMADR4 ;: MEM.LAST ADDRESS,BLK#4
\$ETEND:
.MEXIT

::*****

.SBTTL BEGIN OF AREA TESTED (+20) WHEN PROGRAM RELOCATES.

.=500
BEGIN: MOV PC,SP ;SET UP STACK POINTER
TST -(SP) ;TO EQUAL BEGIN ADDRESS
MOV #ENDPROG,R4 ;PUT END OF PROG ADDRESS INTO R4
MOV SP,@#SAVR6 ;SAVE SP FOR FUTURE USE
MOV #PWRDN,@#24 ;PREPARE FOR FUTURE POWER DOWN
JSR PC,CLRMM ;CLEAR MEM MGMT.

291	000526	005037	000310		CLR	@#TYPCNT	
292	000532	105737	000170		TSTB	@#REL	:RELOCATED ?
293	000536	100002			BPL	4\$:BR IF NOT
294	000540	000167	001344		JMP	TSTSIZ	
295	000544	005067	002146		4\$: CLR	LDFLG	
296	000550	012737	000566	000004	MOV	#7\$,@#4	:PREPARE FOR NON-EXISTANT
297	000556	012737	000014	177746	MOV	#14,@#177746	:11/60 CACHE CONTROL REG
298	000564	000402			BR	6\$:BR IF ALL OK
299	000566	062706	000004		7\$: ADD	#4,SP	:RECOVER FROM TRAP
300	000572	012701	100000		6\$: MOV	#100000,R1	:2ND 16K
301	000576	005021			CLR	(R1)+	:CLEAR FIRST LOC
302	000600	005011			CLR	(R1)	:CLEAR SECOND LOC
303	000602	012777	020004	002120	MOV	#20004,@CSRADR	:DIAG BIT
304	000610	005741			TST	-(R1)	:READ THE LOC TO GET CHECKBITS IF ANY
305	000612	032777	007740	002110	BIT	#7740,@CSRADR	:IF SET WE'RE IN MS11K
306	000620	001007			BNE	5\$:BR IF IN MEM UNDER TEST
307	000622	042767	020000	002060	BIC	#20000,ECCDIS	:JUST IN CASE WE ARE TESTING
308	000630	042767	020000	002054	BIC	#20000,DIAGA	:THE SECOND CSR (172134)
309	000636	000533			BR	SETSWR	:BR IF NOT IN MEM UNDER TEST
310	000640	005077	002064		5\$: CLR	@CSRADR	:CLEAR CSR
311	000644	012767	000001	002044	MOV	#1,LDFLG	:INDICATE PROG IN MEM UNDER TEST
312	000652	052767	020000	002030	BIS	#20000,ECCDIS	:DISABLE ECCINH IN LOW 16K
313	000660	052767	020000	002024	BIS	#20000,DIAGA	:SAME FOR DIAG CHECK
314	000666	005001			CLR	R1	:
315	000670	005000			CLR	R0	:IT IS, CHECK FOR ERRORS
316	000672	005067	002030		CLR	INHREL	
317	000676	005077	002026		1\$: CLR	@CSRADR	:DON'T WANT TO TRAP
318	000702	005711			TST	(R1)	:READ THE LOC
319	000704	032777	100000	002016	BIT	#100000,@CSRADR	:CHECK FOR DOUBLE ERROR
320	000712	001403			BEQ	8\$	
321	000714	004767	013326		JSR	PC,FATERR	:*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
322	000720	000001			1		:*****ERROR NUMBER 1*****
323							
324	000722	005767	002000		8\$: TST	INHREL	:ANY ERRORS FOUND YET ?
325	000726	001012			BNE	2\$:YES, DON'T LOOK ANY FURTHER
326	000730	020127	000500		CMP	R1,#BEGIN	:
327	000734	101007			BHI	2\$:
328	000736	032777	000020	001764	BIT	#20,@CSRADR	:LOOKING FOR SBE'S IN 0-500
329	000744	001403			BEQ	2\$:BR IF NO ERRORS
330	000746	012767	000001	001752	MOV	#1,INHREL	:INHIBIT RELOCATION
331	000754	022701	016154		2\$: CMP	#ENDPROG,R1	:END OF PROGRAM YET?
332	000760	003403			BLE	3\$:BR IF YES
333	000762	062701	000004		ADD	#4,R1	:NO POINT TO NEXT LOCATION
334	000766	000743			BR	1\$:KEY READING
335	000770	005767	001732		3\$: TST	INHREL	:RELOCATION ALLOWED ?
336	000774	001415			BEQ	ONEPAS	:BR IF YES
337	000776	004767	013462		JSR	PC,TPCRLF	:PRINT ROUTINE
338	001002	047516	051040	046105	.ASCIZ	/NO RELOC-SBE IN 0-500/	
339	001010	041517	051455	042502			
340	001016	044440	020116	026460			
341	001024	030065	000060				
342							
343	001030	005000			ONEPAS: .EVEN	CLR	R0
344	001032	005737	000404		TST	@#STESTN	:INITIALIZE POINTER
345	001036	001402			BEQ	TSTRP	:IS THIS THE FIRST PASS
346	001040	000167	000062		JMP	SETSWR	:BR IF YES (TEST TRAP CATCHERS) :SET UP SWREG

```

347 001044 012704 016154      TSTRP: MOV      #ENDPROG,R4      ;ADDRESS OF END OF PROGRAM
348 001050 012700 000377      MOV      #377,R0                ;
349 001054 005001                CLR      R1                      ;POINT TO ADDRESS 0
350 001056 012124                1$:    MOV      (R1)+,(R4)+       ;SAVE 0000 TO BEGIN-
351 001060 020127 000400      CMP      R1,#$MAIL              ;BEGINNING OF ETABLE
352 001064 103774                BLO     1$                       ;BR IF NOT DONE
353 001066 005741                3$:    TST      -(R1)            ;ADJUST POINTER TO TRAP VECTORS
354 001070 010011                4$:    MOV      R0,(R1)          ;WRITE 1'S AND 0'S
355 001072 020011                CMP      R0,(R1)              ;NOW COMPARE
356 001074 001403                BEQ     5$                       ;BR IF OK
357 001076 004767 013144      JSR     PC,FATERR              ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
358 001102 000002                2                                           ;*****ERROR NUMBER 2*****
359
360
361 001104 000300                5$:    SWAB     R0                ;
362 001106 001370                BNE     4$                       ;BR IF BOTH BYTES NOT TESTED
363 001110 005701                TST     R1                       ;HAVE WE REACHED THE BOTTOM YET
364 001112 001365                BNE     3$                       ;BRANCH IF NO
365 001114 012701 000400      MOV      #$MAIL,R1             ;APT, RESTORE TO $MAIL
366 001120 014441                6$:    MOV      -(R4),-(R1)       ;HERE!
367 001122 005701                TST     R1                       ;FINISHED?
368 001124 001375                BNE     6$                       ;BR IF NOT
369 001126 005077 001576      SETSWR: CLR     @CSRADR          ;CLEAR CSR
370 001132 012700 000006      MOV      #6,R0                 ;ADDRESS OF PSW FOR TIMEOUT TRAP
371 001136 012710 000340      MOV      #340,(R0)            ;SET UP TIME OUT TRAP PSW
372 001142 012740 001170      MOV      #2$,-(R0)           ;AND RETURN ADDRESS
373 001146 105737 000420      TSTB   @#$ENV                 ;RUNNING UNDER APT
374 001152 001403                BEQ     1$                       ;IF NOT, SET UP ADDRESS FOR SWR
375 001154 012737 000422 002732 1$:    MOV      #$SWREG,@#SWR        ;OTHERWISE, PREPARE TO USE APT SWR
376 001162 005777 001544      TST     @SWR                   ;DOES SWITCH REG POINTED TO BY SWR EXIST
377 001166 000410                BR      APTSIZ                 ;YES, GO TO APTSIZ
378 001170 062706 000004      2$:    ADD      #4,SP                ;RESTORE STACK POINTER
379 001174 012737 000176 002732  MOV      #SWREG,@#SWR         ;PLACE ADDRESS OF SWITCH REG DESIGNED
380 001202 012737 000174 002734  MOV      #DSPREG,@#DISPLAY    ;FOR COMPUTERS NOT HAVING HARDWARE
381                                           ;SWITCH REG AND RUNNING STAND ALONE.
382                                           ;R4=# END PROG ON EXIT
383                                           ;
384                                           ;
385
386
387
388
389
390
391 001210 012703 000336      APTSIZ: MOV      #MAXMEM,R3      ;POINT R3 TO MAXMEM
392 001214 013737 000326 000332  MOV      @#HIGHTWO,@#$HIMAX    ;IN CASE NO SELF SIZING DONE
393 001222 013737 000330 000334  MOV      @#HIGHADD,@#$MAXM    ;
394 001230 105737 000421      TSTB   @#$ENV                 ;DOES APT ALLOW SELF SIZING
395 001234 100021                BPL     TRYSR                  ;BR IF YES
396 001236 012701 000451      MOV      #SMTYP4+4,R1         ;POINT R1 TO BLOCK TYPE 4+4
397 001242 162701 000004      1$:    SUB      #4,R1                ;POINT TO NEXT BLOCK TYPE
398 001246 105711                TSTB   (R1)                   ;IS IT NON-ZERO
399 001250 001006                BNE     2$                       ;BR IF YES (MEMORY EXISTS)
400 001252 020127 000431      CMP      R1,#$SMTYP1         ;ALL APT BLOCK TYPES BEEN CHECKED
401 001256 101371                BHI     1$                       ;BR IF NO
402 001260 004767 012762      JSR     PC,FATERR              ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT

```

:APTSIZ-THIS ROUTINE WILL SEARCH THE APT MEMORY ETABLE
:AND WHEN A NON ZERO TYPE IS FOUND WILL SETUP TO TEST TO
:GIVEN HIGH ADDRESS. IF APT DEFINES SIZE LO ADDRESS MUST -00000


```

403 001264 000003          3          ;*****ERROR NUMBER 3*****
404
405
406 001266 004767 013722      2$:   JSR   PC,GETADR      ;SET MAX APT ADDR INTO $MAXM +$HIMAX
407 001272 004767 013716      JSR   PC,GETADR      ;ALSO INTO HIGHADD + HIGHTWO
408 001276 000564          BRTPSZ: BR   TYPSIZ    ;TYPE SIZE OF MEM UNDER TEST
409
410 001300 032777 000100 001424 TRYSR: BIT   #100,@SWR      ;USER DEFINED BOUNDARIES
411 001306 001160          BNE   TYPSIZ          ;BR IF YES (DON'T SIZE MEM)
412
413 001310 005067 001400      SLFSIZ: CLR  MSYES      ;INITIALIZE MSYES
414 001314 012703 000324      MOV   #LOWADD,R3     ;GET READY FOR LOW TEST ADDRESS
415 001320 005001          CLR   R1              ;FIRST TEST LOC (0)
416 001322 012710 001514      MOV   #4$, (R0)      ;SET UP RETURN FROM TIMEOUT TRAP
417 001326 005077 001376      1$:   CLR   @CSRADR
418 001332 011146          MOV   (R1),-(SP)     ;SAVE THE CONTENTS OF TEST LOC.
419 001334 016146 000002      MOV   2(R1),-(SP)
420 001340 005011          CLR   (R1)           ;TO SET CHECK BITS OF MS11K
421 001342 005061 000002      CLR   2(R1)
422 001346 016777 001340 001354 MOV   DIAGA,@CSRADR  ;DIAGNOSTIC BIT
423 001354 005711          TST   (R1)           ;READ TO GET CHECK BITS
424 001356 017702 001346      MOV   @CSRADR,R2    ;GET CSR CONTENTS
425 001362 005077 001342      CLR   @CSRADR
426 001366 012661 000002      MOV   (SP)+,2(R1)   ;RESTORE THE HI WORD
427 001372 012611          MOV   (SP)+,(R1)    ;RESTORE THE LO WORD
428 001374 042702 170037      BIC   #170037,R2    ;ONLY CARE ABOUT 11-5
429 001400 005702          TST   R2
430 001402 001440          BEQ   3$
431 001404 005767 001304      TST   MSYES          ;BR IF NO MS11K
432 001410 001032          BNE   22$           ;MS11K-FIRST SUCCESSFUL TEST
433 001412 020127 100000      CMP   R1,#100000    ;NO, BRANCH
434 001416 001015          BNE   11$           ;IF THIS IS FIRST READABLE LOC
435 001420 105737 000272      TSTB  @#MMAVA       ;THEN FIRST ADDRESS IS 0000, ELSE BR
436 001424 001012          BNE   11$           ;MEM MGMT AVAIL ?
437 001426 005767 001264      TST   LDFLG         ;YES, GO PUT ADDR AWAY
438 001432 001407          BEQ   11$           ;RUNNING IN MEM UNDER TEST
439 001434 005037 000324      CLR   @#LOWADD      ;BR IF NOT
440 001440 005037 000322      CLR   @#LOWTWO
441 001444 005267 001244      INC   MSYES
442 001450 000404          BR    2$
443 001452 004767 013446      11$:  JSR   PC,PUTADR     ;CLEAR THESE LOCATIONS
444 001456 005267 001232      INC   MSYES          ;INDICATE SOME MS11K FOUND
445 001462 010137 000320      2$:   MOV   R1,@#MINMEM ;CONTINUE TO FIND HIGH LIMIT
446 001466 010167 001230      MOV   R1,SAVMIN     ;PLACE ADDR AWAY (START ADDR)
447 001472 062703 000012      ADD   #12,R3        ;INDICATE SOME MS11 FOUND
448 001476 062701 020000      22$:  ADD   #20000,R1     ;FIRST TEST ADDRESS
449 001502 000711          BR    1$            ;SAVE FOR RECOVERY FROM RELOC
450 001504 005767 001204      3$:   TST   MSYES        ;POINT TO #MAXMEM
451 001510 001772          BEQ   22$           ;NEXT 4K BOUNDARY
452 001512 000430          BR    7$            ;GO TEST IT
453 001514 062706 000004      4$:   ADD   #4,SP        ;FOUND ANY MS11K YET
454 001520 004767 013214      JSR   PC,MEMMNG     ;BR IF NO
455 001524 105737 000272      TSTB  @#MMAVA       ;RESTORE STACK POINTER AFTER TRAP
456 001530 001421          BEQ   7$            ;SERVICE MM IF AVAILABLE
457 001532 012710 001544      5$:   MOV   #6$, (R0)    ;SEE IF MM HAS TO BE TESTED
458 001536 012701 040000      MOV   #40000,R1    ;BR IF NO MM
                          ;SET UP RETURN ADDRESS FROM TRAP
                          ;BEGIN CHECKING ABOVE 28K

```

```

459 001542 000671          BR      1$
460 001544 062706 000004    6$:    ADD      #4,SP
461 001550 022701 160000    CMP      #160000,R1
462
463
464 001554 001007          BNE      7$
465 001556 022737 007400 172352    CMP      #7400,@#172352
466 001564 001403          BEQ      7$
467 001566 004767 013324    JSR      PC,UPMM
468 001572 000757          BR       5$
469
470
471 001574 024341          7$:    CMP      -(R3),-(R1)
472
473 001576 020127 037776          CMP      R1,#37776
474 001602 001014          BNE      8$
475 001604 005721          TST      (R1)+
476 001606 004767 013312    JSR      PC,PUTADR
477 001612 162713 000002    SUB      #2,(R3)
478 001616 162703 000004    SUB      #4,R3
479 001622 004767 013276    JSR      PC,PUTADR
480 001626 162713 000002    SUB      #2,(R3)
481 001632 000406          BR       TYPsiz
482 001634 004767 013264    8$:    JSR      PC,PUTADR
483
484 001640 162703 000004    SUB      #4,R3
485 001644 004767 013254    JSR      PC,PUTADR
486
487
488 001650 012710 000352    TYPsiz: MOV     #BUSER,(R0)
489 001654 005737 000406    TST     @#$PASS
490 001660 001035          BNE     SETSTK
491 001662 004767 012576    JSR     PC,TPCRLF
492 001666 051503 020122 042101 .ASCIZ  /CSR ADDR = /
493 001674 051104 036440 000040
494
495 001702 012703 002664    .EVEN
496 001706 005023          MOV     #DATBUF,R3
497 001710 016713 001014    CLR     (R3)+
498 001714 105237 000310    MOV     CSRADR,(R3)
499 001720 004767 012700    INCB   @#TYPcnt
500 001724 010403          JSR     PC,TYPCT
501 001726 012701 000322    MOV     R4,R3
502 001732 004767 012514    MOV     #LOWTWO,R1
503 001736 004767 012652    JSR     PC,PCRLF
504
505 001742 004767 012416    JSR     PC,OCTTYP
506 001746 000055          TYPMEM: .ASCIZ 1-1
507
508 001750 004767 012640    .EVEN
509
510 001754 012703 000326    JSR     PC,OCTTYP
511 001760 004767 013244    SETSTK: MOV    #HIGHTWO,R3
512
513 001764 010401          JSR     PC,$GTSIZ
514 001766 062704 000022    MOV     R4,R1
4$:    ADD     #18.,R4

```

```

:RESTORE STACK POINTER
:IF R1 DID NOT READ THE HIGHEST
:LOCATION POINTED TO BY PAR6,
:IT HAS REACHED MAX MS11K.
:GO TO 7$ IF DONE
:CHECK PAR5 FOR MAX
:BR IF DONE
:NOT DONE UPDATE MEM MGMT.

```

```

:CAUSE R3 TO POINT TO $MAXM AND
:R1 TO MAX AVAILABLE MEMORY
:MEANS WE MOVED INTO PAR2
:IF WE DID WE MUST ADJUST ADDRESS
:RESTORE R1
:PUT AWAY ORIGINAL ADDRESS
:ADJUST BITS 15-0
:POINT TO HIGHADD
:PUT IT AWAY
:ADJUST
:AND CONTINUE TO TYPE THE SIZE
:PLACE ADDR IN R1 AT LOCATIONS
:$MAXM AND $HIMAX
:POINT TO HIGHADD
:PLACE ADDR IN R1 AT HIGHADD & HIGHTWO

```

```

:SET UP VECTOR FOR FUTURE TRAP
:ONLY CARE ABOUT FIRST PASS
:BR IF NOT
:PRINT ROUTINE

```

```

:NEED A LOCATION TO PRINT FROM
:CLEAR HIGH ORDER BITS
:ADDRESS TO BE TYPED
:ONE WORD TO BE TYPED
:TYPE THE WORD
:POINT R3 TO LOWEST AVAIL MEM
:TYPE <CR><LF>
:TYPE LOW TEST ADDRESS
:(LOWTWO + LOWADD)

```

```

:TYPE '-'
:TYPE HIGHEST TEST ADDRESS
:(HIGHTWO + HIGHADD)
:POINT R3 TO HIGH ORDER BITS OF HIGH MEM
:GET THE BITS 13-17 ON TOP ADDRESS
:PLACED IN BITS 0-4 OF R2
:#ENDPROG
:APPEND THE ERROR STACK FOR MEMORY

```



```

571 002200 101001          BHI      9$          ;IF SO, GO TO 9$
572 002202 011305          MOV      (R3),R5      ;MODIFY R5
573 002204 020405          9$:     CMP      R4,R5      ;IS LOW LIMIT LOWER THAN HIGH LIMIT
574 002206 103403          BLO     10$
575 002210 004767 012032  JSR     PC,FATERR    ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
576 002214 000005          5          ;*****ERROR NUMBER 5*****
577
578
579
580 002216 012703 000340  10$:    MOV     #SAVMAX,R3
581 002222 011343          MOV     (R3),-(R3)   ;RESTORE CONTENTS OF MAXMEM
582 002224 062713 000002  MEMTST: ADD    #2,(R3) ;MAKE THE CONTENTS OF MAXMEM=
583                                     ;MAX AVAIL MEM+2
584 002230 005725          TST    (R5)+        ;SET R5=MAXMEM +2
585
586
587                                     ;INIT BAKPAT AND SWAPAT
588                                     ;AND CONTINUE TO CLEAR ACTIVE
589                                     ;CHUNK OF MEMORY UNDER TEST
590
591 002232 012702 000312  CLRMEM: MOV    #BAKPAT,R2
592 002236 012212          MOV    (R2)+,(R2)   ;COPY IT INTO SWAPAT
593 002240 000312          SWAB   (R2)         ;SWAP THE BYTES
594 002242 017702 000464  MOV    @SWR,R2      ;GET BITS IN SW REG
595 002246 042702 177740  BIC    #177740,R2   ;ONLY WANT THE TEST NUMBER FOR LOOPING
596 002252 022702 000021  CMP    #21,R2       ;CAN'T BE HIGHER THAN 20
597 002256 101003          BHI    1$           ;BR IF OK
598 002260 004767 011762  JSR    PC,FATERR    ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
599 002264 000006          6          ;*****ERROR NUMBER 6*****
600
601
602 002266 013701 000320  1$:     MOV    @#MINMEM,R1 ;GET FIRST TEST LOC IN SLICE
603 002272 004767 013202  2$:     JSR    PC,TSTADD   ;CHECK FOR ERROR FREE TEST ADDRESS
604 002276 005703          TST    R3           ;R3 = 0 MEANS ERROR FREE LOC FOUND
605 002300 001446          BEQ    4$           ;BR AND SAVE IT
606 002302 016777 000402 000420  3$:     MOV    ECCDIS,@CSRADR ;ECC DISABLE
607 002310 062701 000004  ADD    #4,R1        ;NEXT WORD
608 002314 020105          CMP    R1,R5        ;END OF SLICE YET ?
609 002316 103765          BLO   2$           ;BR IF NOT
610 002320 004767 012140  JSR    PC,TPCRLF    ;PRINT ROUTINE
611 002324 047516 042440 051122  .ASCII /NO ERROR FREE LOC FOR ECC TESTS/
612 002332 051117 043040 042522
613 002340 020105 047514 020103
614 002346 047506 020122 041505
615 002354 020103 042524 052123
616 002362 123
617 002363 015 041012 043505  .ASCIZ <15><12>/BEGINNING AT TST13/
618 002370 047111 044516 043516
619 002376 040440 020124 051524
620 002404 030524 000063
621
622 002410 012702 000013  .EVEN
623 002414 000402          MOV    #13,R2      ;START AT TST13
624 002416 010137 000320  BR     CONT
625 002422 016777 000262 000300  4$:     MOV    R1,@#MINMEM ;SET UP TEST LOC
626 002430 010500          MOV    ECCDIS,@CSRADR ;INHIBIT ECC
          MOV    R5,R0

```



```
627 002432 005040          5$: CLR    -(R0)          ;BEGIN CLEARING FROM TOP
628 002434 020004          CMP    R0,R4          ;TO BOTTOM
629 002436 101375          BHI    5$             ;
630 002440 005077 000264   CLR    @CSRADR        ;RESTORE CSR
631
632                          ;ENTER HERE FROM TSTSCP ROUTINE AT END OF EACH TEST
633
634 002444 005037 000302   CLR    @#PASFLG       ;INIT SOFTEST PASS FLAG
635 002450 110237 000404   MOVB  R2,@#$TESTN     ;SET UP $TESTN WITH THE TEST
636 002454 110277 000254   MOVB  R2,@DISPLAY     ;NUMBER TO BE EXECUTED & DISPLAYED
637
638 002460 005077 000244   LOOP: CLR @CSRADR ;
639 002464 032777 004000 000240 BIT    #4000,@SWR     ;ENABLE PRINTOUT OF SBE'S(INH ECC)
640 002472 001403          BEQ    1$             ;BR IF NO
641 002474 016777 000210 000226 MOV    ECCDIS,@CSRADR ;YES, INHIBIT ECC
642 002502 010401          1$:  MOV    R4,R1         ;LOWEST ADDRESS UNDER TEST
643 002504 010246          MOV    R2,-(SP)      ;SAVE R2
644 002506 012703 000376   MOV    #376,R3       ;POINT R3 TO SCRATCH STACK
645 002512 004767 012406   JSR    PC,PUTADR     ;GEN 18 BIT ADDRESS FROM R1
646                          ;AND STORE IT IN (R3) AND (R3-2)
647 002516 005743          TST    -(R3)         ;POINT R3 TO HIGH ORDER BITS
648 002520 004767 012504   JSR    PC,$GTSIZ     ;PLACE BITS 13-17 OF ADDRESS BITS
649                          ;INTO 0-4 OF R2.
650 002524 010400          MOV    R4,R0         ;PLACE ADDRESS OF LOWEST LOC
651                          ;UNDER TEST IN R0
652 002526 010401          MOV    R4,R1         ;AND INTO R1
653 002530 010403          MOV    R4,R3         ;AND INTO R3
654 002532 012602          MOV    (SP)+,R2      ;RESTORE R2
655 002534 006302          ASL    R2             ;DOUBLE IT
656 002536 060702          ADD    PC,R2
657 002540 066207 000004   ADD    TBL-,(R2),PC  ;GO TO TEST # STORED IN BITS
658                          ;3-0 OF SWITCH REG.
659 002544 000172          TBL:  TST0-TBL       ;RELATIVE ADDRESS OF TEST 0
660 002546 000326          TST1-TBL             ;RELATIVE ADDRESS OF TEST 1
661 002550 000620          TST2-TBL             ;RELATIVE ADDRESS OF TEST 2
662 002552 001000          TST3-TBL             ;RELATIVE ADDRESS OF TEST 3
663 002554 001232          TST4-TBL             ;RELATIVE ADDRESS OF TEST 4
664 002556 001406          TST5-TBL             ;RELATIVE ADDRESS OF TEST 5
665 002560 002056          TST6-TBL             ;RELATIVE ADDRESS OF TEST 6
666 002562 002776          TST7-TBL             ;RELATIVE ADDRESS OF TEST 7
667 002564 003772          TST10-TBL            ;RELATIVE ADDRESS OF TEST 10
668 002566 004420          TST11-TBL            ;RELATIVE ADDRESS OF TEST 11
669 002570 005052          TST12-TBL            ;RELATIVE ADDRESS OF TEST 12
670 002572 005512          TST13-TBL            ;RELATIVE ADDRESS OF TEST 13
671 002574 005624          TST14-TBL            ;RELATIVE ADDRESS OF TEST 14
672 002576 005736          TST15-TBL            ;RELATIVE ADDRESS OF TEST 15
673 002600 006570          TST16-TBL            ;RELATIVE ADDRESS OF TEST 16
674 002602 006740          TST17-TBL            ;RELATIVE ADDRESS OF TEST 17
675 002604 007076          RELOC-TBL           ;RELATIVE ADDRESS OF RELOC
676
677                          ;SCOPE ROUTINE
678                          ;
679                          ;PROG COMES HERE AFTER EACH TEST AND
680                          ;IF CTRL C GO TO ERROR HISTORY TYPEOUT.
681                          ;IF SR=2000 (BIT 10) THEN HALT
682                          ;IF SR=40000 (BIT 14) THEN LOOP ON TEST
                          ;DEFINED BY SR BITS <3:0>, ELSE GO ON
```

```
683                                     :          TO NEXT TEST.
684                                     :
685 002606 005077 000116   TSTSCP: CLR      @CSRADR ;
686 002612 105737 000420   TSTB      @#SENV      ;RUNNING UNDER APT?
687 002616 001002         BNE      CNTSCP      ;IF SO GO TO CNTSCP
688 002620 004767 013252   JSR      PC,CHECKC   ;TEST FOR CTRL-C AND IF TYPED
689                                     ;GO TO ERROR HISTORY AND HALT
690 002624 113702 000404   CNTSCP: MOVB   @#$TESTN,R2 ;PUT TEST NUMBER IN R2 LO BYTE
691 002630 005237 000410   INC      @#$DEVCT    ;TELL APT EVERYTHING'S OK
692 002634 032777 002000 000070   BIT     #2000,@SWR   ;HALT AFTER EACH TEST?
693 002642 001401         BEQ     TSTGO        ;BRANCH IF NOT HALTING
694 002644 000000         SWHALT: HALT
695 002646 032777 040000 000056   TSTGO: BIT     #40000,@SWR ;LOOP ON TEST DESIRED?
696 002654 001301         BNE     LOOP        ;YES, GO START SAME TEST
697 002656 105202         INCB   R2
698 002660 000167 177536   JMP     CONT        ;GO CONTINUE EXECUTING NEXT TEST
699
700 002664 000000 000000   DATBUF: .WORD  0,0
701 002670 000000 000000   TSTDAT: .WORD  0,0
702 002674 000000 000000   SBEMSK: .WORD  0,0
703 002700 000000 000000   DBEMSK: .WORD  0,0
704 002704 000000 000000   DATSAV: .WORD  0,0
705 002710 000002         ECCDIS: .WORD  2
706 002712 000004         DIAGA: .WORD  4
707 002714 000000         MSYES: .WORD  0
708 002716 000000         LDFLG: .WORD  0
709 002720 000000         BASE: .WORD  0
710 002722 100000         SAVMIN: .WORD 100000 ;OFFSET FOR RELOC
711 002724 000000         ERRFLG: .WORD 0 ;TO SAVE MINMEM FROM RELOC
712 002726 000000         INHREL: .WORD 0 ;FLAG TO INDICATE FIRST ERROR FOR HEADER
713 002730 172136         CSRADR: 172136 ;FLAG TO INDICATE NO RELOCATION ALLOWED
714                                     ;DEFAULT CSR ADDRESS
715
716
717
718
719
720
721
722
723 002732 177570   SWR: 177570 ;CHANGES TO SWREG IF NO HARDWARE SWITCH REGISTER
724 002734 177570   DISPLAY:177570 ;CHANGES TO DSPREG IF NO HARDWARE DISPLAY REG
725
726                                     ;*(1) THIS TEST ASSUMES THAT THE MEMORY IS IN A STATE
727                                     ;* OF ALL 0'S AND R0 HAS THE ADDRESS OF THE LOWEST
728                                     ;* LOCATION UNDER TEST
729                                     ;*(2) IT CHECKS FOR PROPER BANK SELECTION BY WRITING
730                                     ;* 1'S IN A LOCATION AND CHECKING FOR 0'S IN THE SAME
731                                     ;* LOCATIONS OF OTHER 4K BANKS OF THE MEMORY
732                                     ;* [I.E. LOCATIONS LIKE 7766 AND 27766 ETC.]
733
734 002736 122737 000000 000404   TSTO: CMPB   #0,@#$TESTN ;CHECK FOR PROPER TEST SEQUENCE
735 002744 001403         BEQ     +10
736 002746 004767 011274         JSR     PC,SEQERR ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
737 002752 000007         7 ;*****ERROR NUMBER 7*****
738
```



```

739 002754 012703 177777      MOV      #177777,R3
740 002760 010401      1$:     MOV      R4,R1      ;R1 = ADDRESS OF LOWEST LOCATION OF MEMORY UNDER TEST
741 002762 010310      MOV      R3,(R0)     ;SET ALL THE BITS AT (R0)
742 002764 020001      2$:     CMP      R0,R1      ;IS R0 POINTING TO THE SAME MEMORY LOCATION AS R1
743 002766 001417      BEQ      4$          ;IN WHICH CASE CHECK FOR ALL 1'S AT (R1)
744 002770 005711      TST      (R1)        ;OTHERWISE CHECK (R1) FOR ALL 0'S
745 002772 001430      BEQ      5$
746 002774 020311      CMP      R3,(R1)     ;IF R1 IS NOT EQUAL TO R0 AND (R1)
747                                     ;DOES NOT CONTAIN ALL 0'S THEN
748                                     ;CHECK TO SEE IF (R0) = (R1)
749 002776 001004      BNE      3$
750 003000 012767 000010 000042      MOV      #10,12$     ;*****ERROR NUMBER 10*****
751
752 003006 000403      BR       10$
753 003010
754 003010 012767 000011 000032      3$:     MOV      #11,12$     ;*****ERROR NUMBER 11*****
755
756 003016 010046      10$:    MOV      R0,-(SP)     ;SAVE R0 ON STACK
757 003020 105237 000275      INCB    @#$ADERR     ;AN ADDRESSING ERROR IS SUSPECTED
758 003024 000407      BR       11$
759 003026 020311      4$:     CMP      R3,(R1)     ;CHECK (R1) FOR ALL 1'S
760 003030 001411      BEQ      5$
761 003032 012767 000012 000010      MOV      #12,12$     ;*****ERROR NUMBER 12*****
762
763 003040 010046      MOV      R0,-(SP)     ;SAVE R0 ON STACK
764 003042 010300      MOV      R3,R0
765 003044 004767 010470      11$:    JSR      PC,ERROR    ;GO TO THE ERROR SUBROUTINE
766 003050 000000      12$:    .WORD
767 003052 012600      MOV      (SP)+,R0     ;ERROR NUMBER TO BE REPORTED WILL BE PLACED HERE
768                                     ;RESTORE R0
769 003054 013706 000346      5$:     MOV      @#SAVR6,SP   ;RESTORE THE STACK POINTER
770 003060 062701 020000      ADD     #20000,R1     ;CAUSE R1 TO POINT TO THE SAME CHIP
771                                     ;LOCATION IN THE NEXT 4K BANK OF MEMORY
772                                     ;BY ADDING 1 TO THE 14TH BIT OF ADDRESS IN R1
773 003064 020105      CMP      R1,R5       ;COMPARE R1 WITH THE HIGHEST MEMORY
774                                     ;LOCATION WHICH IS STORED IN R5
775 003066 103736      BLO     2$           ;IF R1 LESS THAN R5 THEN REPEAT THE TEST FROM 2$
776
777 003070 000646      ENDO:   BR       TSTSCP
778                                     ;INITIAL DATA TEST
779                                     ;THIS TEST CHECKS THE DI/DO LINES BY
780                                     ;SHIFTING A 1 THROUGH THE WORD.
781
782 003072 122737 000001 000404      TST1:   CMPB    #1,@#$TESTN ;CHECK FOR PROPER SEQUENCE
783 003100 001403      BEQ     .+10         ;BR IF OK
784 003102 004767 011140      JSR     PC,SEQERR    ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
785 003106 000013      13     ;*****ERROR NUMBER 13*****
786
787 003110 013701 000320      MOV     @#MINMEM,R1  ;GET TEST ADDRESS
788 003114 012767 000001 177542      1$:     MOV     #1,DATBUF    ;SET THE FIRST TEST BIT
789 003122 005067 177540      CLR     DATBUF+2     ;CLEAR 2ND WORD
790 003126 016711 177532      2$:     MOV     DATBUF,(R1)  ;WRITE TEST WORD 1
791 003132 016761 177530 000002      MOV     DATBUF+2,2(R1) ;AND TEST WORD 2
792 003140 026711 177520      CMP     DATBUF,(R1)  ;NOW READ THEM
793 003144 001405      BEQ     4$           ;BR IF FIRST 16 OK
794 003146 016700 177512      MOV     DATBUF,R0    ;GET GOOD DATA FOR ERROR MSG

```

```

795 003152 004767 010362 JSR PC,ERROR ;*ERROR* REPORT ERROR MESSAGE
796 003156 000014 14 ;*****ERROR NUMBER 14*****
797
798
799 003160 026761 177502 000002 4$: CMP DATBUF+2,2(R1) ;NOW READ SECOND WORD
800 003166 001405 BEQ 5$ ;BR IF OK
801 003170 016700 177472 MOV DATBUF+2,R0 ;GOOD DATA FOR ERROR MSG
802 003174 004767 010340 JSR PC,ERROR ;*ERROR* REPORT ERROR MESSAGE
803 003200 000015 15 ;*****ERROR NUMBER 15*****
804
805
806
807 003202 005767 177460 5$: TST DATBUF+2 ;HAS LAST BIT BEEN TESTED ?
808 003206 100404 BMI 6$ ;MINUS MEANS BIT 31
809 003210 004567 010140 JSR R5,DASHL ;NO, SHIFT TEST BIT LEFT
810 003214 002664 .WORD DATBUF ;IN DATBUF
811 003216 000743 BR 2$ ;GO WRITE NEW TEST DATA
812
813 ;NOW GOING TO SHIFT A 0 IN DATA DIRECTION
814 003220 012767 177776 177436 6$: MOV #177776,DATBUF ;PUT A 0 IN BIT 0
815 003226 012767 177777 177432 MOV #-1,DATBUF+2 ;AND 1'S IN ALL OTHERS
816 003234 012702 002664 MOV #DATBUF,R2 ;LOC OF WRITE DATA
817 003240 066702 177454 ADD BASE,R2 ;OFFSET IT IN CASE OF RELOC
818 003244 016711 177414 66$: MOV DATBUF,(R1) ;WRITE THE DATA
819 003250 016761 177412 000002 MOV DATBUF+2,2(R1) ;2 WORDS WORTH
820 003256 026711 177402 CMP DATBUF,(R1) ;NOW READ FIRST WORD
821 003262 001405 BEQ 7$ ;BR IF OK
822 003264 016700 177374 MOV DATBUF,R0 ;GOOD DATA
823 003270 004767 010244 JSR PC,ERROR ;*ERROR* REPORT ERROR MESSAGE
824 003274 000016 16 ;*****ERROR NUMBER 16*****
825
826
827 003276 026761 177364 000002 7$: CMP DATBUF+2,2(R1) ;NOW, READ SECOND WORD
828 003304 001405 BEQ 8$ ;BR IF OK
829 003306 016700 177354 MOV DATBUF+2,R0 ;GOOD DATA
830 003312 004767 010222 JSR PC,ERROR ;*ERROR* REPORT ERROR MESSAGE
831 003316 000017 17 ;*****ERROR NUMBER 17*****
832
833
834 003320 022762 077777 000002 8$: CMP #77777,2(R2) ;TESTED BIT 31 YET?
835 003326 001410 BEQ 10$ ;BR IF YES, WE'RE DONE
836 003330 006162 000002 ROL 2(R2) ;SHIFT DATBUF
837 003334 006112 ROL (R2) ;AND DATBUF +2
838 003336 103403 BCS 9$ ;
839 003340 005362 000002 DEC 2(R2) ;
840 003344 000261 SEC ;SET CARRY FOR NEXT ROL
841 003346 000736 9$: BR 66$ ;KEEP GOING
842 003350 062701 020000 10$: ADD #20000,R1 ;NEXT 4K BANK
843 003354 020105 CMP R1,R5 ;COMPARE AGAINST HIGHEST VIRTUAL MEM
844 003356 103656 BLO 1$ ;
845
846 003360 000167 177222 END1: JMP TSTSCP
847
848
849 ;TEST2 THIS TEST CHECKS TO SEE THAT EACH ADDRESS
850 ; BIT IN EACH 4K BANK CAN BE ASSERTED UNIQUELY.

```



```

907
908
909 003544 122737 000003 000404 TST3:  CMPB  #3,@#STESTN  ;CHECK FOR PROPER SEQUENCE NUMBER
910 003552 001403  BEQ    1$          ;OK, BRANCH
911 003554 004767 010466  JSR    PC,SEQERR  ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
912 003560 000025  JSR    25          ;*****ERROR NUMBER 25*****
913
914
915 003562 010446 1$:  MOV    R4,-(SP)    ;SAVE R4
916 003564 013702 000320  MOV    @#MINMEM,R2 ;MINMEM IS LOWEST ADDRESS
917 003570 010203  MOV    R2,R3      ;PUT IT IN R3 ALSO
918 003572 062702 000004  ADD    #4,R2      ;POINT R2 TO LAST BYTE +1
919 003576 012713 177777  MOV    #-1,(R3)   ;WRITE ALL ONES IN
920 003602 012763 177777 000002  MOV    #-1,2(R3)  ;THE 4 TEST BYTES
921 003610 105013 2$:  CLRB   (R3)      ;CLEAR A BYTE
922 003612 013701 000320  MOV    @#MINMEM,R1 ;INITIALIZE R1 FOR EACH PASS
923 003616 020201 3$:  CMP    R2,R1     ;IF EQUAL, JUST READ LAST BYTE
924 003620 001456  BEQ    5$          ;BR IF EQUAL
925 003622 020301  CMP    R3,R1     ;IS THIS THE BYTE OF ZEROS
926 003624 001022  BNE    4$          ;BR IF NOT
927 003626 122711 000000  CMPB   #0,(R1)   ;IT IS, COMPARE FOR ZEROS
928 003632 001415  BEQ    6$          ;
929 003634 012700 000377  MOV    #377,R0    ;SET UP LO BYTE
930 003640 032701 000001  BIT    #1,R1      ;WHICH BYTE ARE WE TESTING ?
931 003644 001001  BNE    9$          ;LO BYTE R0 IS OK
932 003646 000300  SWAB   R0         ;HIGH BYTE, MAKE R0 RIGHT
933 003650 010146 9$:  MOV    R1,-(SP)   ;SAVE R1
934 003652 042701 000001  BIC    #1,R1      ;FOR PRINTOUT
935 003656 004767 007656  JSR    PC,ERROR   ;*ERROR* REPORT ERROR MESSAGE
936 003662 000026  JSR    26          ;*****ERROR NUMBER 26*****
937
938 003664 012601  MOV    (SP)+,R1   ;RESTORE R1 FOR FURTHER TESTING
939 003666 005201 6$:  INC    R1         ;NEXT BYTE
940 003670 000752  BR     3$         ;RETURN
941 003672 122711 177777 4$:  CMPB   #-1,(R1)  ;ITS NOT THE BYTE OF 0'S, READ 1'S
942 003676 001425  BEQ    7$         ;
943 003700 010304  MOV    R3,R4     ;GET ADDRESS
944 003702 042704 177775  BIC    #177775,R4 ;TO FIND BYTE
945 003706 030104  BIT    R1,R4     ;IS IT SET IN TEST ADDRESS ?
946 003710 001003  BNE    12$       ;NO, BRANCH CAUSE IT'S NOT IN SAME WORD
947 003712 012700 177777  MOV    #-1,R0    ;DATA SHOULD BE ALL ONES
948 003716 000406  BR     10$       ;GO REPORT ERROR
949 003720 012700 000377 12$: MOV    #377,R0   ;IT'S IN SAME WORD, BUT WHICH BYTE ?
950 003724 032701 000001  BIT    #1,R1     ;CHECK FOR LO BYTE
951 003730 001401  BEQ    10$       ;NO, LEAVE R0 ALONE
952 003732 000300  SWAB   R0        ;SWAP BYTES
953 003734 010146 10$: MOV    R1,-(SP)  ;SAVE R1
954 003736 042701 000001  BIC    #1,R1     ;FOR PRINTOUT
955 003742 004767 007572  JSR    PC,ERROR   ;*ERROR* REPORT ERROR MESSAGE
956 003746 000027  JSR    27          ;*****ERROR NUMBER 27*****
957
958 003750 012601  MOV    (SP)+,R1   ;RESTORE R1
959 003752 005201 7$:  INC    R1         ;MOVE TO NEXT BYTE
960 003754 000720  BR     3$         ;
961 003756 112713 177777 5$:  MOVB   #-1,(R3)  ;RESTORE 1'S TO BYTE JUST TESTED
962 003762 005203  INC    R3         ;INC TO NEXT BYTE

```



```

1019
1020 004152 122737 000005 000404 TST5:  CMPB    #5,@#STESTN    ;CHECK FOR PROPER TEST SEQUENCE
1021 004160 001403                BEQ      1$
1022 004162 004767 010060        JSR      PC,SEQERR    ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
1023 004166 000033                33              ;*****ERROR NUMBER 33*****
1024
1025 004170 005077 176534        1$:      CLR      @CSRADR
1026 004174 012767 000001 176462 T5A:    MOV      #1,DATBUF    ;INITIAL DATA
1027 004202 005067 176460        CLR      DATBUF+2    ;32 BITS WORTH
1028 004206 012767 000001 176460 20$:    MOV      #1,SBEMSK   ;INITIAL ERROR MASK
1029 004214 005067 176456        CLR      SBEMSK+2    ;32 BITS WORTH
1030 004220 016767 176440 176442 30$:    MOV      DATBUF,TSTDAT
1031 004226 016767 176434 176436        MOV      DATBUF+2,TSTDAT+2 ;TO SAVE ORIG DATA
1032 004234 105737 000302        TSTB    @#PASFLG    ;COMP DATA ON SECOND PASS ONLY
1033 004240 001404                BEQ      40$         ;BR IF FIRST PASS
1034 004242 005167 176422        COM     TSTDAT      ;SECOND PASS, COMP BOTH WORDS
1035 004246 005167 176420        COM     TSTDAT+2
1036 004252 016702 176412        40$:    MOV      TSTDAT,R2
1037 004256 016703 176410        MOV     TSTDAT+2,R3
1038 004262 012767 002670 006522        MOV     #TSTDAT,SOURCE ;SET UP ADDRESS FOR CHKGEN
1039 004270 004767 006570        JSR     PC,CHKGEN   ;GEN CHECKBITS ON TSTDAT
1040 004274 004567 007104        JSR     R5,BITCOM   ;BIT COMPLEMENT ROUTINE
1041 004300 002674                .WORD   SBEMSK      ;MASK
1042 004302 002670                .WORD   TSTDAT     ;DATA
1043 004304 013701 000320        50$:    MOV     @#MINMEM,R1  ;FIRST TEST ADDRESS
1044 004310 016777 176374 176412        MCV     ECCDIS,@CSRADR ;FORCE VALIDATE WITH ECC DISABLED
1045 004316 016721 176346        MOV     TSTDAT,(R1)+ ;WRITE FIRST 16 BITS
1046 004322 016700 006466        MOV     CHECK,R0    ;GET CHECKBITS FROM CHKGEN
1047 004326 056700 176360        BIS     DIAGA,R0    ;SET DIAGNOSTIC CHECK A
1048 004332 010077 176372        MOV     R0,@CSRADR ;LOAD CSR WITH IMAGE IN R0
1049 004336 016711 176330        MOV     TSTDAT+2,(R1) ;WRITE SECOND 16 BITS AND
1050                                ;CHECK BITS. WE NOW HAVE CHECKBITS
1051                                ;GENERATED ON DATBUF AND DATA WITH
1052                                ;ONE BIT IN ERROR (AS PER SBEMSK).
1053 004342 016777 176342 176360        MOV     ECCDIS,@CSRADR ;DISABLE ERROR CORRECTING
1054 004350 016700 176314        MOV     TSTDAT,R0   ;SET UP GOOD DATA FOR ERROR REPT.
1055 004354 020041                CMP     R0,-(R1)    ;READ THE LOW WORD
1056 004356 001403                BEQ     55$         ;BR IF OK
1057 004360 004767 007154        JSR     PC,ERROR    ;*ERROR* REPORT ERROR MESSAGE
1058 004364 000034                34      ;*****ERROR NUMBER 34*****
1059
1060 004366 016700 176300        55$:    MOV     TSTDAT+2,R0 ;HIGH WORD
1061 004372 062701 000002        ADD     #2,R1      ;POINT R1 TO SECOND 16 BITS
1062 004376 020011                CMP     R0,(R1)    ;READ THE HIGH WORD
1063 004400 001403                BEQ     56$         ;BR IF OK
1064 004402 004767 007132        JSR     PC,ERROR    ;*ERROR* REPORT ERROR MESSAGE
1065 004406 000035                35      ;*****ERROR NUMBER 35*****
1066
1067 004410 005077 176314        56$:    CLR     @CSRADR    ;ENABLE ECC
1068 004414 010200                MOV     R2,R0      ;THIS IS ORIGINAL DATA
1069 004416 020041                CMP     R0,-(R1)    ;SEE IF ITS BEEN CORRECTED
1070 004420 001403                BEQ     57$         ;IT SHOULD HAVE BEEN
1071 004422 004767 007112        JSR     PC,ERROR    ;*ERROR* REPORT ERROR MESSAGE
1072 004426 000036                36      ;*****ERROR NUMBER 36*****
1073

```



```

1074 004430 032777 000020 176272 57$: BIT #20,@CSRADR ;CHECK IF SBE INDICATE SET
1075 004436 001011 BNE 58$ ;BR IF IT IS SET
1076 004440 010146 MOV R1,-(SP) ;SAVE R1
1077 004442 016701 176262 MOV CSRADR,R1 ;GET CSR ADDRESS
1078 004446 012700 000020 MOV #20,R0 ;READY R0 FOR PRINTOUT
1079 004452 004767 007062 JSR PC,ERROR ;*ERROR* REPORT ERROR MESSAGE
1080 004456 000037 37 ;*****ERROR NUMBER 37*****
1081
1082 004460 012601 MOV (SP)+,R1 ;RESTORE R1
1083 004462 005077 176242 58$: CLR @CSRADR ;RESTORE CSR
1084 004466 010300 MOV R3,R0 ;HIGH WORD
1085 004470 062701 000002 ADD #2,R1 ;POINT TO HIGH WORD
1086 004474 020011 CMP R0,(R1) ;SEE IF ITS BEEN CORRECTED
1087 004476 001403 BEQ 59$ ;BR IF OK
1088 004500 004767 007034 JSR PC,ERROR ;*ERROR* REPORT ERROR MESSAGE
1089 004504 000040 40 ;*****ERROR NUMBER 40*****
1090
1091 004506 032777 000020 176214 59$: BIT #20,@CSRADR ;SBE BIT SET ?
1092 004514 001010 BNE 60$ ;BR IF YES
1093 004516 010146 MOV R1,-(SP) ;SAVE R1
1094 004520 016701 176204 MOV CSRADR,R1 ;GET ADDRESS OF CSR BEING TESTED
1095 004524 012700 000020 MOV #20,R0 ;FOR PRINTOUT
1096 004530 004767 007004 JSR PC,ERROR ;*ERROR* REPORT ERROR MESSAGE
1097 004534 000041 41 ;*****ERROR NUMBER 41*****
1098
1099 004536 005737 000406 60$: TST @#$PASS
1100 004542 001425 BEQ ENDS ;Q V ONLY ONE ITERATION
1101 004544 005767 176126 TST SBEMSK+2 ;TEST FOR LAST MASK BIT
1102 004550 100404 BMI 65$ ;MINUS MEANS BIT 31
1103 004552 004567 006576 JSR R5,DASHL ;SHIFT LEFT ROUTINE FOR 32 BITS
1104 004556 002674 .WORD SBEMSK ;WORD TO BE SHIFTED
1105 004560 000617 BR 30$
1106 004562 005767 176100 65$: TST DATBUF+2 ;LAST DATA BIT ?
1107 004566 100404 BMI 70$ ;WHICH IS BIT 31
1108 004570 004567 006560 JSR R5,DASHL ;SHIFT IT
1109 004574 002664 .WORD DATBUF ;
1110 004576 000603 BR 20$
1111 004600 105737 000302 70$: TSTB @#PASFLG ;FIRST OR SECOND PASS ?
1112 004604 001004 BNE ENDS ;NON ZERO MEANS WE'RE DONE
1113 004606 105237 000302 INCB @#PASFLG ;NOT DONE, GO DO SECOND PASS
1114 004612 000167 177356 JMP T5A
1115 004616 000167 175764 ENDS: JMP TSTSCP
1116 ;SINGLE BIT ERROR TEST TO INSURE THAT A WRITE
1117 ;BYTE CLEARS SINGLE BIT ERRORS.
1118 ;
1119 004622 122737 000006 000404 TST6: CMPB #6,@$TESTN ;CHECK FOR PROPER TEST SEQUENCE
1120 004630 001403 BEQ 1$
1121 004632 004767 007410 JSR PC,SEQERR ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
1122 004636 000042 42 ;*****ERROR NUMBER 42*****
1123
1124 004640 010701 1$: MOV PC,R1 ;GET PC
1125 004642 012700 005534 MOV #END6,R0 ;END OF THIS TEST
1126 004646 066700 176046 ADD BASE,R0 ;FOR OFFSET
1127 004652 005711 2$: TST (R1) ;READ A LOC
1128 004654 032777 000020 176046 BIT #20,@CSRADR ;LOOKING FOR SBE'S IN THIS TEST
1129 004662 001403 BEQ 3$ ;BR IF NO ERROR

```

1130	004664	005267	000650		INC	T6FLG	:INDICATE WITH FLAG
1131	004670	000404			BR	4\$:AND BRANCH
1132	004672	062701	000004	3\$:	ADD	#4,R1	:MOVE TO NEXT WORD
1133	004676	020100			CMP	R1,R0	:END OF TEST 6 YET ?
1134	004700	103764			BLO	2\$:NO, BRANCH
1135	004702	004467	006420	4\$:	JSR	R4,SAVOT4	:SAVE REGS R0 TO R4
1136	004706	005077	176016		CLR	@CSRADR	:CLEAR CSR
1137	004712	012767	000001	175744	MOV	#1,DATBUF	:INITIAL DATA
1138	004720	005067	175742		CLR	DATBUF+2	:32 BITS WORTH
1139	004724	012767	000001	175742	T6A: MOV	#1,SBEMSK	:INITIAL ERROR MASK
1140	004732	005067	175740		CLR	SBEMSK+2	:32 BITS WORTH
1141	004736	016767	175722	175724	T6B: MOV	DATBUF,TSTDAT	:SAVE ORIGINAL DATA
1142	004744	016767	175716	175720	MOV	DATBUF+2,TSTDAT+2	:BOTH WORDS
1143	004752	016767	175706	175724	MOV	DATBUF,DATSAV	:IN CASE PROG HAS SBE
1144	004760	016767	175702	175720	MOV	DATBUF+2,DATSAV+2	:DITTO
1145	004766	012767	002670	006016	MOV	#TSTDAT,SOURCE	:NEED ADDRESS FOR CHKGEN
1146	004774	004767	006064		JSR	PC,CHKGEN	:GENERATE CHECK BITS
1147	005000	004567	006400		JSR	R5,BITCOM	:BIT COM ROUTINE
1148	005004	002674			.WORD	SBEMSK	:MASK FOR COMPLEMENTING
1149	005006	002670			.WORD	TSTDAT	:WORD TO BE COMPLEMENTED
1150	005010	013701	000320	30\$:	MOV	@MINMEM,R1	:FIRST TEST ADDRESS
1151	005014	010104			MOV	R1,R4	:PUT IT IN R4 ALSO
1152	005016	016777	175666	175704	MOV	ECCDIS,@CSRADR	:INHIBIT ECC
1153	005024	016724	175640		MOV	TSTDAT,(R4)+	:WRITE 16 BITS
1154	005030	016703	005760		MOV	CHECK,R3	:GET CHECKBITS
1155	005034	056703	175652		BIS	DIAGA,R3	:SET DIAGNOSTIC A BIT
1156	005040	010377	175664		MOV	R3,@CSRADR	:LOAD CSR WITH NEW IMAGE
1157	005044	016714	175622		MOV	TSTDAT+2,(R4)	:WRITE HIGH WORD+CHECKBITS
1158	005050	005077	175654		CLR	@CSRADR	:IT'S DANGEROUS IF WE DON'T
1159	005054	012702	002674		MOV	#SBEMSK,R2	:ADDRESS OF ERROR MASK
1160	005060	066702	175634		ADD	BASE,R2	:
1161	005064	162704	000002		SUB	#2,R4	:ADJUST ADDRESS
1162	005070	112714	177777	40\$:	MOVB	#-1,(R4)	:WRITE A BYTE OF 1'S
1163	005074	132712	177777		BITB	#-1,(R2)	:
1164	005100	001476			BEQ	60\$:
1165	005102	005767	000432		TST	T6FLG	:SINGLE ERRORS IN TST6 AREA ?
1166	005106	001016			BNE	45\$:BR IF YES
1167	005110	005077	175614		CLR	@CSRADR	:CLEAR CSR
1168	005114	105714			TSTB	(R4)	:READ
1169	005116	032777	000020	175604	BIT	#20,@CSRADR	:SINGLE ERROR INDICATE SET ?
1170	005124	001453			BEQ	50\$:
1171	005126	016701	175576		MOV	CSRADR,R1	:FOR PRINTOUT
1172	005132	005000			CLR	R0	:
1173	005134	004767	006400		JSR	PC,ERROR	:*ERROR* REPORT ERROR MESSAGE
1174	005140	000043			43		:*****ERROR NUMBER 43*****
1175							
1176	005142	000444			BR	50\$:CONTINUE
1177	005144	010446		45\$:	MOV	R4,-(SP)	:NEED A SCRATCH LOC
1178	005146	163716	000320		SUB	@MINMEM,(SP)	:TO FIND BYTE OFFSET
1179	005152	012703	002704		MOV	#DATSAV,R3	:GET ADDRESS
1180	005156	066703	175536		ADD	BASE,R3	:OFFSET FOR RELOC
1181	005162	062603			ADD	(SP)+,R3	:ADD BYTE COUNT
1182	005164	112713	177777		MOVB	#-1,(R3)	:WRITE BYTE OF 1'S LIKE TEST LOCATION
1183	005170	016777	175516	175532	MOV	DIAGA,@CSRADR	:DIAG BIT
1184	005176	105714			TSTB	(R4)	:READ THE BYTE TO GET THE CHECKBITS
1185							

1186	005200	017703	175524		MOV	@CSRADR,R3	:GET CHECKBITS
1187	005204	042703	170037		BIC	#170037,R3	:CLEAR OTHER BITS
1188	005210	005077	175514		CLR	@CSRADR	:
1189	005214	012767	002704	005570	MOV	#DATSAV,SOURCE	:FOR CHKGEN
1190	005222	004767	005636		JSR	PC,CHKGEN	:GENERATE CHECKBITS
1191	005226	020367	005562		CMP	R3,CHECK	:COMPARE ACTUAL VS. EXPECTED
1192	005232	001410			BEQ	50\$:BR IF OK
1193	005234	010300			MOV	R3,R0	:GOOD DATA
1194	005236	012701	013014		MOV	#CHECK,R1	:ADDRESS OF CHECKBITS
1195	005242	066701	175452		ADD	BASE,R1	:PLUS OFFSET
1196	005246	004767	006266		JSR	PC,ERROR	:ERROR ROUTINE
1197	005252	000043			43		
1198	005254	122714	177777	50\$:	CMPB	#-1,(R4)	:CHECK DATA
1199	005260	001461			BEQ	70\$:BR IF OK
1200	005262	010401			MOV	R4,R1	:GET R4
1201	005264	042701	000001		BIC	#1,R1	:TO MAKE EVEN ADDR FOR PRINTOUT
1202	005270	004767	006244		JSR	PC,ERROR	:*ERROR* REPORT ERROR MESSAGE
1203	005274	000044			44		:*****ERROR NUMBER 44*****
1204							
1205	005276	005767	000236	60\$:	TST	T6FLG	:SINGLE ERRORS IN TST6 AREA ?
1206	005302	001011			BNE	65\$:BR IF YES
1207	005304	105714			TSTB	(R4)	:READ THE BYTE
1208	005306	032777	000020	175414	BIT	#20,@CSRADR	:SBE ERROR BIT SET ?
1209	005314	001357			BNE	50\$:SHOULD BE SET, BR IF OK
1210	005316	004767	006216		JSR	PC,ERROR	:*ERROR* REPORT ERROR MESSAGE
1211	005322	000045			45		:*****ERROR NUMBER 45*****
1212							
1213	005324	000437			BR	70\$:CONTINUE
1214	005326	010446		65\$:	MOV	R4,-(SP)	:SAVE R4
1215	005330	163716	000320		SUB	@#MINMEM,(SP)	:FOR BYTE OFFSET
1216	005334	012703	002704		MOV	#DATSAV,R3	:ADDRESS OF DATA
1217	005340	066703	175354		ADD	BASE,R3	:FOR RELOC
1218	005344	062603			ADD	(SP)+,R3	:BYTE OFFSET
1219	005346	112713	177777		MOVB	#-1,(R3)	:WRITE A BYTE OF 1'S
1220	005352	016777	175334	175350	MOV	DIAGA,@CSRADR	:DIAG BIT
1221	005360	105714			TSTB	(R4)	:READ THE BYTE
1222	005362	017746	175342		MOV	@CSRADR,-(SP)	:GETCONTENTS
1223	005366	005077	175336		CLR	@CSRADR	:CLEAR THE CSR
1224	005372	042716	170037		BIC	#170037,(SP)	:ONLY WANT CHECKBITS
1225	005376	012767	002704	005406	MOV	#DATSAV,SOURCE	:FOR CHKGEN
1226	005404	004767	005454		JSR	PC,CHKGEN	:GENERATE CHECKBITS ON DATA
1227	005410	022667	005400		CMP	(SP)+,CHECK	:COMPARE ACTUAL VS. EXPECTED
1228	005414	001403			BEQ	70\$:BR IF OK
1229	005416	004767	006116		JSR	PC,ERROR	:ERROR ROUTINE
1230	005422	000045			45		
1231	005424	132712	177777	70\$:	BITB	#-1,(R2)	:CHECK FOR LAST BYTE
1232	005430	001012			BNE	80\$:
1233	005432	005202			INC	R2	:
1234	005434	005204			INC	R4	:MOVE TO NEXT BYTE
1235	005436	013701	000320		MOV	@#MINMEM,R1	:FIRST TEST ADDRESS
1236	005442	032704	000002		BIT	#2,R4	:TEST FOR LOWER WORD
1237	005446	001610			BEQ	40\$:BR IF IT'S LOW 16 BITS
1238	005450	062701	000002		ADD	#2,R1	:ADJUST POINTER FOR ERROR REPT.
1239	005454	000605			BR	40\$	
1240	005456	005737	000406	80\$:	TST	@#SPASS	:FIRST PASS ?
1241	005462	001420			BEQ	100\$:BR IF YES

1242	005464	005767	175206		TST	SBEMSK+2		;LAST ERROR BIT ?
1243	005470	100405			BMI	90\$;MINUS MEANS BIT 31
1244	005472	004567	005656		JSR	R5,DASHL		;DOUBLE ARITHMETIC SHIFT LEFT
1245	005476	002674			.WORD	SBEMSK		;ON THE ERROR MASK
1246	005500	000167	177232		JMP	T6B		
1247	005504	005767	175156	90\$:	TST	DATBUF+2		;LAST DATA BIT ?
1248	005510	100405			BMI	100\$;MINUS = BIT 31
1249	005512	004567	005636		JSR	R5,DASHL		
1250	005516	002664			.WORD	DATBUF		;SHIFT IT LEFT
1251	005520	000167	177200		JMP	T6A		
1252	005524	004767	005610	100\$:	JSR	PC,RSTOT4		;RESTORE REGS R0 TO R4
1253	005530	005067	000004		CLR	T6FLG		;FOR NEXT TIME
1254	005534	000167	175046	END6:	JMP	TSTSCP		
1255	005540	000000		T6FLG:	.WORD	0		
1256								
1257								;DOUBLE BIT ERROR FORCE TO CHECK DOUBLE ERROR
1258								;STATUS BIT AND TRAP THROUGH LOCATION 114.
1259								
1260	005542	122737	000007	000404	TST7:	CMPE	#7,@#STESTN	
1261	005550	001403			BEQ	1\$		
1262	005552	004767	006470		JSR	PC,SEQERR		;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
1263	005556	000046			46			;*****ERROR NUMBER 46*****
1264								
1265	005560	005077	175144	1\$:	CLR	@CSRADR		;CLEAR CSR
1266	005564	013701	000320		MOV	@#MINMEM,R1		
1267	005570	004467	005532		JSR	R4,SAVOT4		;SAVE REGS R0 TO R4
1268	005574	005067	175064	10\$:	CLR	DATBUF		;MAKE INITIAL DATA
1269	005600	005067	175062		CLR	DATBUF+2		;ALL ZEROS
1270	005604	012767	000001	175062	20\$:	MOV	#1,SBEMSK	;INITIAL SINGLE ERROR MASK
1271	005612	005067	175060		CLR	SBEMSK+2		;SECOND WORD
1272	005616	012767	000001	175054	30\$:	MOV	#1,DBEMSK	;INITIAL DOUBLE ERROR MASK
1273	005624	005067	175052		CLR	DBEMSK+2		;32 BITS HERE ALSO
1274	005630	016767	175030	175032	35\$:	MOV	DATBUF,TSTDAT	
1275	005636	016767	175024	175026		MOV	DATBUF+2,TSTDAT+2	
1276	005644	105737	000302		TSTB	@#PASFLG		;NO COMPLEMENTING FIRST PASS
1277	005650	001404			BEQ	40\$		
1278	005652	005167	175012		COM	TSTDAT		;COMP FIRST WORD
1279	005656	005167	175010		COM	TSTDAT+2		;SECOND WORD
1280	005662	005077	175042	40\$:	CLR	@CSRADR		
1281	005666	026767	175002	175004		CMP	SBEMSK,DBEMSK	;CAN'T HAVE THE SAME ERROR BIT SET
1282	005674	001004			BNE	45\$;IN BOTH MASKS
1283	005676	026767	174774	174776		CMP	SBEMSK+2,DBEMSK+2	;COULD BE EQUAL IN SECOND WORD
1284	005704	001502			BEQ	65\$;GO MAKE THEM NOT EQUAL
1285	005706	012767	002670	005076	45\$:	MOV	#TSTDAT,SOURCE	;SOURCE ADDRESS FOR CHKGEN
1286	005714	004767	005144		JSR	PC,CHKGEN		;GO GENERATE CHECK BITS
1287	005720	004567	005460		JSR	R5,BITCOM		;FORCE SINGLE ERROR
1288	005724	002674			.WORD	SBEMSK		;AS PER SBEMSK
1289	005726	002670			.WORD	TSTDAT		;ON DATA IN TSTDAT
1290	005730	004567	005450		JSR	R5,BITCOM		;FORCING DOUBLE ERROR
1291	005734	002700			.WORD	DBEMSK		;THIS MASK INDICATES SECOND BIT IN ERROR
1292	005736	002670			.WORD	TSTDAT		;SAME DATA WORD
1293	005740	016704	005050		MOV	CHECK,R4		;GET CHECKBITS
1294	005744	056704	174742		BIS	DIAGA,R4		;SET DIAGNOSTIC A BIT
1295	005750	016777	174734	174752	MOV	ECCDIS,@CSRADR		;INHIBIT ECC
1296	005756	016721	174706		MOV	TSTDAT,(R1)+		;WRITE 16 BITS
1297	005762	010477	174742		MOV	R4,@CSRADR		;LOAD CSR

1298	005766	016711	174700		MOV	TSTDAT+2,(R1)	:WRITE HIGH WORD
1299	005772	005077	174732		CLR	@CSRADR	:CLEAR CSR AGAIN
1300	005776	162701	000002		SUB	#2,R1	:ADJUST TEST ADDRESS
1301	006002	005711			TST	(R1)	:READ THE LOCATION
1302	006004	032777	100000	174716	BIT	#100000,@CSRADR	:LOOK FOR DBE STATUS BIT
1303	006012	001011			BNE	50\$:IT SHOULD BE SET
1304	006014	010146			MOV	R1,-(SP)	:SAVE R1
1305	006016	016701	174706		MOV	CSRADR,R1	:GET CSR ADDRESS
1306	006022	012700	100000		MOV	#100000,R0	:SHOW BIT 15 SET
1307	006026	004767	005506		JSR	PC,ERROR	:*ERROR* REPORT ERROR MESSAGE
1308	006032	000047			47		:*****ERROR NUMBER 47*****
1309							
1310	006034	012601			MOV	(SP)+,R1	:RESTORE R1
1311	006036	012702	006100	50\$:	MOV	#60\$,R2	:SET UP FOR BUS PBL
1312	006042	066702	174652		ADD	BASE,R2	:
1313	006046	010237	000114		MOV	R2,@#114	:
1314	006052	052777	000001	174650	BIS	#1,@CSRADR	:SET DOUBLE ERROR INDICATE
1315							
1316	006060	005711			TST	(R1)	:READ THE TEST LOCATION AGAIN
1317	006062	005077	174642		CLR	@CSRADR	:CLEAR STATUS REG
1318	006066	005000			CLR	R0	:CLEAR R0
1319	006070	004767	005444		JSR	PC,ERROR	:*ERROR* REPORT ERROR MESSAGE
1320	006074	000050			50		:*****ERROR NUMBER 50*****
1321							
1322	006076	000402			BR	61\$:IF ERROR, DON'T ADJUST STACK
1323	006100	062706	000004	60\$:	ADD	#4,SP	:ADJUST STACK FROM TRAP
1324	006104	005737	000406	61\$:	TST	@#PASS	:FIRST PASS ?
1325	006110	001424			BEQ	80\$:BR IF YES
1326	006112	005767	174564	65\$:	TST	DBEMSK+2	:CHECK MASK FOR LAST BIT
1327	006116	100404			BMI	70\$:MINUS = BIT31
1328	006120	004567	005230		JSR	R5,DASHL	:DOUBLE SHIFT
1329	006124	002700			.WORD	DBEMSK	
1330	006126	000640			BR	35\$	
1331	006130	005767	174542	70\$:	TST	SBEMSK+2	:CHECK SINGLE ERROR MASK TOO
1332	006134	100404			BMI	75\$:BR IF DONE
1333	006136	004567	005212		JSR	R5,DASHL	:SHIFT
1334	006142	002674			.WORD	SBEMSK	:
1335	006144	000624			BR	30\$	
1336	006146	105737	000302	75\$:	TSTB	@#PASFLG	:FIRST PASS
1337	006152	001003			BNE	80\$:NON ZERO MEANS WE'RE DONE
1338	006154	105237	000302		INCB	@#PASFLG	:FIRST PASS, NOT DONE
1339	006160	000605			BR	10\$:KEEP GOING
1340	006162	005737	000406	80\$:	TST	@#PASS	:CHECKING PASS AGAIN
1341	006166	001406			BEQ	82\$:CHECK 1K ADDR FUNCTION ON
1342							:FIRST PASS ONLY
1343	006170	016777	174514	174532	MOV	ECCDIS,@CSRADR	:OTHERWISE DISABLE ECC
1344	006176	005021			CLR	(R1)+	:CLEAR THE DBE'S
1345	006200	005011			CLR	(R1)	:
1346	006202	000545			BR	90\$:AND EXIT
1347	006204	005077	174520	82\$:	CLR	@CSRADR	:CLEAR CSR
1348	006210	005002			CLR	R2	:OUR COUNTER
1349	006212	005711			TST	(R1)	:READ THE LOCATION
1350	006214	017703	174510		MOV	@CSRADR,R3	:MOV CHECKBITS INTO R3
1351	006220	005077	174504		CLR	@CSRADR	:CLEAR CSR AGAIN
1352	006224	042703	170037		BIC	#170037,R3	:ONLY WANT CHECKBITS
1353	006230	006303			ASL	R3	:POSITION THEM

```

1354 006232 006303      ASL      R3          ;INTO LOW BYTE
1355 006234 006303      ASL      R3
1356 006236 000303      SWAB     R3          ;HERE
1357 006240 004767 007144 JSR      PC,GET1K    ;GET THE 1K BANK
1358 006244 020300      CMP      R3,R0      ;BETTER BE THE SAME
1359 006246 001413      BEQ      85$        ;BR IF OK
1360 006250 010146      MOV      R1,-(SP)    ;SAVE R1 FOR NOW
1361 006252 016777 174432 174450 MOV      ECCDIS,@CSRADR ;DISABLE ECC
1362 006260 010311      MOV      R3,(R1)    ;PUT BAD DATA TO BE PRINTED INTO R1
1363 006262 005077 174442 CLR      @CSRADR     ;RESTORE CSR
1364 006266 004767 005246 JSR      PC,ERROR    ;*ERROR* REPORT ERROR MESSAGE
1365 006272 000051      51             ;*****ERROR NUMBER 51*****
1366
1367 006274 012601      MOV      (SP)+,R1    ;RESTORE R1
1368 006276 016777 174406 174424 85$: MOV      ECCDIS,@CSRADR ;DISABLE ERROR CORRECTING
1369 006304 005021      CLR      (R1)+      ;TO CLEAR ANY DOUBLE ERRORS
1370 006306 005011      CLR      (R1)
1371 006310 005077 174414      CLR      @CSRADR    ;RESTORE CSR
1372 006314 042701 003777 BIC      #3777,R1    ;STRIP AWAY TO LAST 1K
1373 006320 062701 004000 ADD      #4000,R1    ;MOVE UP TO NEXT 1K
1374 006324 020105      CMP      R1,R5      ;OVER TOP YET ?
1375 006326 103073      BHIS     90$        ;BR IF YES
1376 006330 004767 007144      88$: JSR      PC,TSTADD   ;CHECK FOR ERROR FREE LOC
1377 006334 005703      TST      R3          ;ZERO MEANS ERROR FREE
1378 006336 001456      BEQ      86$        ;BR IF OK
1379 006340 005202      INC      R2          ;COUNT ONE WORD CHECKED
1380 006342 020227 001000 CMP      R2,#1000    ;TESTED WHOLE 1K SLICE YET ?
1381 006346 103447      BLO      87$        ;BR IF NOT
1382 006350 004767 006110 JSR      PC,TPCRLF   ;PRINT ROUTINE
1383 006354 051524 033524 047055 .ASCIZ  /TST7-NO ERROR FREE LOC IN 1K SLICE AT /
1384 006362 020117 051105 047522
1385 006370 020122 051106 042505
1386 006376 046040 041517 044440
1387 006404 020116 045461 051440
1388 006412 044514 042503 040440
1389 006420 020124      000
1390
1391 006424 042701 003776      .EVEN
1392 006430 012703 006534 BIC      #3776,R1    ;WANT STARTING ADDRESS OF 1K
1393 006434 066703 174260 MOV      #SAV7+2,R3 ;SCRATCH LOC FOR ADDRESS
1394 006440 010146      ADD      BASE,R3    ;FOR OFFSET
1395 006442 004767 006456 MOV      R1,-(SP)    ;SAVE R1
1396 006446 105277 171636 JSR      PC,PUTADR   ;PUT ADDRESS IN SAV7 & SAV7+2
1397 006452 005741      INCB    @TYPCNT     ;PRINT ONE WORD
1398 006454 004767 006144 TST      -(R1)      ;ADJUST FOR PRINTOUT
1399 006460 005002      JSR      PC,TYPOCT  ;PRINT IT HERE
1400 006462 012601      CLR      R2          ;RESTORE R2 FOR NEXT SLICE
1401 006464 000704      MOV      (SP)+,R1    ;RESTORE R1
1402 006466 062701 000004 87$: BR      85$        ;GO LOOK FOR GOOD LOC
1403 006472 000716      ADD      #4,R1      ;NEXT WORD
1404 006474 016721 174170 86$: BR      88$        ;GO CHECK IT
1405 006500 010477 174224 MOV      TSTDAT,(R1)+ ;WRITE FIRST WORD
1406 006504 016711 174162 MOV      R4,@CSRADR  ;CSR IMAGE WITH BAD CHECKBITS
1407 006510 162701 000002 MOV      TSTDAT+2,(R1) ;WRITE 2ND WORD + CHECKBITS
1408 006514 000633      SUB     #2,R1       ;ADJUST R1
1409 006516 005077 174206 90$: BR      82$        ;CONTINUE TESTING
1409 006516 005077 174206 90$: CLR      @CSRADR    ;CLEAR CSR TO NORMAL

```



```

1410 006522 004767 004612          JSR   PC,RSTOT4      ;RESTORE THE REGS 0 TO 4
1411 006526 000167 174054          END7: JMP   TSTSCP
1412 006532 000000 000000          SAV7: .WORD 0,0
1413
1414
1415
1416
1417
1418 006536 122737 000010 000404 TST10: CMPB  #10,@#$TESTN
1419 006544 001403                    BEQ   1$
1420 006546 004767 005474          JSR   PC,SEQERR      ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
1421 006552 000052                    52      ;*****ERROR NUMBER 52*****
1422
1423 006554 004467 004546          1$:   JSR   R4,SAVOT4  ;SAVE REGS R0 TO R4
1424 006560 005067 174100          10$:  CLR   DATBUF        ;INITIAL DATA
1425 006564 005067 174076          CLR   DATBUF+2      ;2 WORDS WORTH
1426 006570 012767 000001 174076 20$:  MOV   #1,SBEMSK     ;INITIAL ERROR MASK
1427 006576 005067 174074          CLR   SBEMSK+2
1428 006602 012767 000001 174070 30$:  MOV   #1,DBEMSK     ;DOUBLE ERROR MASK
1429 006610 005067 174066          CLR   DBEMSK+2
1430 006614 016767 174044 174046 35$:  MOV   DATBUF,TSTDAT ;PRESERVE ORIG DATA
1431 006622 016767 174040 174042  MOV   DATBUF+2,TSTDAT+2
1432 006630 105737 000302          TSTB  @#PASFLG      ;SECOND PASS YET ?
1433 006634 001404                    BEQ   40$            ;BR IF NO
1434 006636 005167 174026          COM   TSTDAT        ;COMPL DATA ON SECOND PASS
1435 006642 005167 174024          COM   TSTDAT+2
1436 006646 005077 174056          40$:  CLR   @CSRADR
1437 006652 026767 174022 174014  CMP   DBEMSK,SBEMSK ;CHECK FOR SAME MASKS
1438 006660 001004                    BNE   45$            ;BR IF OK
1439 006662 026767 174014 174006  CMP   DBEMSK+2,SBEMSK+2
1440 006670 001476                    BEQ   70$            ;BR IF THEY'RE EQUAL
1441 006672 012767 002670 004112 45$:  MOV   #TSTDAT,SOURCE ;SET UP ADDRESS FOR CHKGEN
1442 006700 004767 004160          JSR   PC,CHKGEN     ;GENERATE CHECK BITS
1443 006704 004567 004474          JSR   R5,BITCOM     ;COMPL ROUTINE
1444 006710 002674                    .WORD SBEMSK         ;MASK
1445 006712 002670                    .WORD TSTDAT        ;DATA
1446 006714 004567 004464          JSR   R5,BITCOM     ;TO MAKE DOUBLE ERROR
1447 006720 002700                    .WORD DBEMSK
1448 006722 002670                    .WORD TSTDAT
1449 006724 016702 004064          MOV   CHECK,R2     ;GET CHECKBITS
1450 006730 056702 173756          BIS   DIAGA,R2     ;SET DIAGNOSTIC A BIT
1451 006734 013701 000320          50$:  MOV   @#MINMEM,R1  ;TEST ADDRESS
1452 006740 016777 173744 173762  MOV   ECCDIS,@CSRADR ;INHIBIT ECC
1453 006746 016721 173716          MOV   TSTDAT,(R1)+ ;WRITE FIRST 16 BITS
1454 006752 010277 173752          MOV   R2,@CSRADR   ;LOAD CSR WITH IMAGE FROM R2
1455 006756 016711 173710          MOV   TSTDAT+2,(R1) ;SECOND 16 BITS+CHECKBITS
1456 006762 105067 004064          CLRB  UPPFLG       ;INDICATE LOWER WORD
1457 006766 013703 000320          MOV   @#MINMEM,R3  ;TEST ADDRESS
1458 006772 005077 173732          55$:  CLR   @CSRADR      ;CLEAR IT
1459 006776 005223                    INC   (R3)+        ;INC INSTRUCTION
1460 007000 026741 173664          CMP   TSTDAT,-(R1) ;CHECK FOR UNCHANGED DATA
1461 007004 001405                    BEQ   60$            ;SHOULD BE UNCHANGED
1462 007006 016700 173656          MOV   TSTDAT,R0    ;FOR PRINTOUT
1463 007012 004767 004522          JSR   PC,ERROR      ;*ERROR* REPORT ERROR MESSAGE
1464 007016 000053                    53      ;*****ERROR NUMBER 53*****
1465

```

```

1466 007020 062701 000002      60$:  ADD      #2,R1      ;POINT TO UPPER WORD
1467 007024 026711 173642      CMP      TSTDAT+2,(R1) ;READ IT
1468 007030 001434              BEQ      80$          ;BR IF UNCHANGED
1469 007032 016700 173634      MOV      TSTDAT+2,R0  ;
1470 007036 004767 004476      JSR      PC,ERROR    ;*ERROR* REPORT ERROR MESSAGE
1471 007042 000054              54          ;*****ERROR NUMBER 54*****
1472
1473 007044 105767 004002      65$:  TSTB     UPPFLG      ;LOWER WORD
1474 007050 001003              BNE     66$          ;BR IF NO
1475 007052 105267 003774      INCB    UPPFLG
1476 007056 000745              BR      55$
1477 007060 005737 000406      66$:  TST     @W$PASS    ;CHECK PASS #
1478 007064 001424              BEQ     90$          ;BR IF FIRST
1479 007066 005767 173610      70$:  TST     DBEMSK+2   ;LAST BIT IN MASK ?
1480 007072 100404              BMI     75$          ;BR IF BIT 31
1481 007074 004567 004254      JSR     R5,DASHL    ;SHIFT ROUTINE
1482 007100 002700              .WORD   DBEMSK
1483 007102 000644              BR      35$
1484 007104 005767 173566      75$:  TST     SBEMSK+2   ;LAST BIT IN SINGLE ERROR MASK ?
1485 007110 100404              BMI     80$          ;BR IF YES
1486 007112 004567 004236      JSR     R5,DASHL    ;SHIFT
1487 007116 002674              .WORD   SBEMSK
1488
1489 007120 000630              BR      30$
1490 007122 105737 000302      80$:  TSTB     @W$PASFLG   ;WHICH PASS
1491 007126 001003              BNE     90$          ;BR IF WE'RE DONE
1492 007130 105237 000302      INCB    @W$PASFLG   ;INDICATE SECOND PASS COMING
1493 007134 000611              BR      10$
1494 007136 016777 173546 173564 90$:  MOV     ECCDIS,@CSRADR ;INHIBIT ECC TO CLEAR DBE'S
1495 007144 005011              CLR     (R1)
1496 007146 005041              CLR     -(R1)
1497 007150 005077 173554      CLR     @CSRADR     ;RESTORE CSR TO NORMAL
1498 007154 004767 004160      JSR     PC,RSTOT4   ;RESTORE THE REGS
1499 007160 000167 173422      END10: JMP     TSTSCP
1500
1501
1502
1503
1504 007164 122737 000011 000404 TST11: CMPB    #11,@W$TESTN
1505 007172 001403              BEQ     1$
1506 007174 004767 005046      JSR     PC,SEQERR   ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
1507 007200 000055              55          ;*****ERROR NUMBER 55*****
1508
1509 007202 004467 004120      1$:   JSR     R4,SAVOT4   ;SAVE REGS R0 TO R4
1510 007206 005067 173452      10$:  CLR     DATBUF      ;INITIAL DATA
1511 007212 005067 173450      CLR     DATBUF+2    ;32 BITS WORTH
1512 007216 012767 000001 173450 20$:  MOV     #1,SBEMSK   ;SINGLE ERROR MASK
1513 007224 005067 173446      CLR     SBEMSK+2
1514 007230 012767 000001 173442 30$:  MOV     #1,DBEMSK   ;DOUBLE ERROR MASK
1515 007236 005067 173440      CLR     DBEMSK+2
1516 007242 016767 173416 173420 35$:  MOV     DATBUF,TSTDAT ;PRESERVE ORIG DATA
1517 007250 016767 173412 173414  MOV     DATBUF+2,TSTDAT+2
1518 007256 105737 000302      TSTB    @W$PASFLG   ;WHICH PASS ?
1519 007262 001404              BEQ     40$          ;FIRST PASS, NO COMPLEMENTING
1520 007264 005167 173400      COM     TSTDAT
1521 007270 005167 173376      COM     TSTDAT+2    ;SECOND PASS, COMPLEMENT TSTDAT

```


1522	007274	005077	173430		40\$:	CLR	@CSRADR	:	
1523	007300	026767	173370	173372		CMP	SBEMSK,DBEMSK	:	:CHECK FOR SAME MASKS
1524	007306	001004				BNE	45\$:	:BR IF NOT EQUAL
1525	007310	026767	173362	173364		CMP	SBEMSK+2,DBEMSK+2	:	:SECOND WORD ALSO
1526	007316	001473				BEQ	70\$:	:BR TO MAKE THEM NOT EQUAL
1527	007320	012767	002670	003464	45\$:	MOV	#TSTDAT,SOURCE	:	:ADDRESS FOR CHKGEN
1528	007326	004767	003532			JSR	PC,CHKGEN	:	:GO GENERATE CHECK BITS
1529	007332	004567	004046			JSR	R5,BITCOM	:	:BIT COMP ROUTINE
1530	007336	002674				.WORD	SBEMSK	:	:MASK
1531	007340	002670				.WORD	TSTDAT	:	:DATA WORD
1532	007342	004567	004036			JSR	R5,BITCOM	:	:MUST FORCE SECOND ERROR
1533	007346	002700				.WORD	DBEMSK	:	:MASK
1534	007350	002670				.WORD	TSTDAT	:	
1535	007352	016702	003436			MOV	CHECK,R2	:	:GET CHECKBITS
1536	007356	056702	173330			BIS	DIAGA,R2	:	:SET DIAGNOSTIC A BIT
1537	007362	013701	000320		50\$:	MOV	@MINMEM,R1	:	:TEST LOCATION
1538	007366	016777	173316	173334		MOV	ECCDIS,@CSRADR	:	:DISABLE ECC
1539	007374	016721	173270			MOV	TSTDAT,(R1)+	:	:WRITE FIRST 16 BITS
1540	007400	010277	173324			MOV	R2,@CSRADR	:	:LOAD CSR WITH IMAGE FROM R2
1541	007404	016711	173262			MOV	TSTDAT+2,(R1)	:	:WRITE SECOND 16 BITS + CHECKBITS
1542	007410	005077	173314			CLR	@CSRADR	:	:CLEAR CSR
1543	007414	013702	000320			MOV	@MINMEM,R2	:	:GET ADDRESS OF TEST LOC
1544	007420	010203				MOV	R2,R3	:	:R2 DESIGNATES FIRST BYTE
1545	007422	062703	000003			ADD	#3,R3	:	:R3 DESIGNATES LAST BYTE
1546	007426	112722	000360		55\$:	MOV	#360,(R2)+	:	:TRY WRITING A BYTE
1547	007432	013701	000320			MOV	@MINMEM,R1	:	
1548	007436	026711	173226			CMP	TSTDAT,(R1)	:	:CHECK FOR UNCHANGED DATA
1549	007442	001405				BEQ	60\$:	:BR IF OK
1550	007444	016700	173220			MOV	TSTDAT,R0	:	:FOR ERROR MSG
1551	007450	004767	004064			JSR	PC,ERROR	:	:*ERROR* REPORT ERROR MESSAGE
1552	007454	000056				56		:	:*****ERROR NUMBER 56*****
1553								:	
1554	007456	062701	000002		60\$:	ADD	#2,R1	:	:UPPER WORD
1555	007462	026711	173204			CMP	TSTDAT+2,(R1)	:	:READ SECOND WORD
1556	007466	001405				BEQ	65\$:	:BR IF UNCHANGED
1557	007470	016700	173176			MOV	TSTDAT+2,R0	:	
1558	007474	004767	004040			JSR	PC,ERROR	:	:*ERROR* REPORT ERROR MESSAGE
1559	007500	000057				57		:	:*****ERROR NUMBER 57*****
1560								:	
1561	007502	020203			65\$:	CMP	R2,R3	:	:TESTED LAST BYTE ?
1562	007504	001350				BNE	55\$:	:BR IF NO
1563	007506	005737	000406		70\$:	TST	@\$PASS	:	:FIRST PASS ?
1564	007512	001424				BEQ	100\$:	:BR IF YES
1565	007514	005767	173162			TST	DBEMSK+2	:	:CHECKING FOR LAST ERROR BIT
1566	007520	100404				BMI	80\$:	:BR IF DONE HERE
1567	007522	004567	003626			JSR	R5,DASHL	:	:NOT DONE, SHIFT LEFT
1568	007526	002700				.WORD	DBEMSK	:	:DOUBLE ERROR MASK
1569	007530	000644				BR	35\$:	
1570	007532	005767	173140		80\$:	TST	SBEMSK+2	:	:LAST SBE MASK
1571	007536	100404				BMI	90\$:	:BR IF DONE WITH THIS PASS
1572	007540	004567	003610			JSR	R5,DASHL	:	:DOUBLE WORD SHIFT ROUTINE
1573	007544	002674				.WORD	SBEMSK	:	
1574	007546	000630				BR	30\$:	
1575	007550	105737	000302		90\$:	TSTB	@\$PASFLG	:	:TEST PASS FLAG
1576	007554	001003				BNE	100\$:	:NON ZERO MEANS WE'RE DONE
1577	007556	105237	000302			INCR	@\$PASFLG	:	:NOT DONR

```

1578 007562 000611          BR      10$
1579 007564 016777 173120 173136 100$: MOV    ECCDIS,@CSRADR ;DISABLE ECC
1580 007572 013701 000320      MOV    @MINMEM,R1 ;TEST LOCATION
1581 007576 005021          CLR    (R1)+
1582 007600 005011          CLR    (R1) ;TO ERASE ANY DBE'S FROM TESTING
1583 007602 005077 173122      CLR    @CSRADR ;RESTORE CSR
1584 007606 004767 003526      JSR    PC,RSTOT4 ;RESTORE REGS R0 TO R4
1585 007612 000167 172770      END11: JMP    TSTSCP
1586
1587 ;DOUBLE BIT ERROR WRITE CANCEL WITH
1588 ;WORD WRITE.
1589 ;CHECKS WRITE INHIBIT WITH WORD WRITES TO
1590 ;WORD WITH DOUBLE ERROR.
1591
1592 007616 122737 000012 000404 TST12: CMPB   #12,@$TESTN ;CHECK FOR PROPER TEST SEQUENCE
1593 007624 001403          BEQ    1$ ;BR IF OK
1594 007626 004767 004414      JSR    PC,SEQERR ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
1595 007632 000060          60 ;*****ERROR NUMBER 60*****
1596
1597 007634 004467 003466          1$: JSR    R4,SAVOT4 ;SAVE REGS R0 TO R4
1598 007640 005067 173020      T12A: CLR    DATBUF ;BACKGROUND FOR DOUBLE ERRORS
1599 007644 005067 173016      CLR    DATBUF+2 ;2 WORDS WORTH
1600 007650 012767 000001 173016      MOV    #1,SBEMSK ;SINGLE ERROR MASK
1601 007656 005067 173014      CLR    SBEMSK+2
1602 007662 012767 000001 173010 T12B: MOV    #1,DBEMSK ;DOUBLE ERROR MASK
1603 007670 005067 173006      CLR    DBEMSK+2
1604 007674 016767 172764 172766 35$: MOV    DATBUF,TSTDAT ;DATA FOR TEST
1605 007702 016767 172760 172762      MOV    DATBUF+2,TSTDAT+2 ;BOTH WORDS
1606 007710 105737 000302      TSTB   @PASFLG ;COMP DATA ON SECOND PASS ONLY
1607 007714 001404          BEQ    40$ ;BR IF FIRST PASS
1608 007716 005167 172746      COM    TSTDAT ;COMP FIRST WORD
1609 007722 005167 172744      COM    TSTDAT+2 ;NOW SECOND WORD
1610 007726 026767 172742 172744 40$: CMP    SBEMSK,DBEMSK ;CHECK FOR IDENTICAL MASKS
1611 007734 001004          BNE    45$ ;BR IF DIFFERENT
1612 007736 026767 172734 172736      CMP    SBEMSK+2,DBEMSK+2 ;UPPER WORD TOO
1613 007744 001501          BEQ    70$ ;BR TO MAKE THEM NOT EQUAL
1614 007746 012767 002670 003036 45$: MOV    #TSTDAT,SOURCE ;NEED ADDR OF DATA FOR CHKGEN
1615 007754 004767 003104      JSR    PC,CHKGEN ;GO GENERATE CHECK BITS
1616 007760 004567 003420      JSR    R5,BITCOM ;BIT COMP ROUTINE TO FORCE ERRORS
1617 007764 002674          .WORD SBEMSK ;FIRST ERROR
1618 007766 002670          .WORD TSTDAT ;DATA
1619 007770 004567 003410      JSR    R5,BITCOM ;CALL IT AGAIN
1620 007774 002700          .WORD DBEMSK ;FOR SECOND ERROR
1621 007776 002670          .WORD TSTDAT
1622 010000 016700 003010      MOV    CHECK,R0 ;GET CHECKBITS
1623 010004 056700 172702      BIS    DIAGA,R0 ;SET DIAGNOSTIC BIT
1624 010010 013701 000320          MOV    @MINMEM,R1 ;FIRST TEST ADDRESS
1625 010014 016777 172670 172706 50$: MOV    ECCDIS,@CSRADR ;INHIBIT ECC
1626 010022 016721 172642      MOV    TSTDAT,(R1)+ ;WRITE FIRST 16 BITS
1627 010026 010077 172676      MOV    R0,@CSRADR ;LOAD CSR FROM R0
1628 010032 016711 172634      MOV    TSTDAT+2,(R1) ;WRITE SECOND 16 BITS + CHECKBITS
1629 010036 105067 003010      CLRB   UPPFLG ;SET FOR 2 LOOPS
1630 010042 162701 000002      SUB    #2,R1 ;POINT TO LOW WORD
1631 010046 005077 172656          CLR    @CSRADR ;CLEAR CSR
1632 010052 012721 177400          MOV    #177400,(R1)+ ;TRY WRITING LOCATION
1633 010056 013701 000320          MOV    @MINMEM,R1

```



```

1634 010062 026711 172602      CMP      TSTDAT,(R1)      ;CHECK FOR ORIGINAL DATA
1635 010066 001405              BEQ      60$             ;SHOULD BE UNCHANGED
1636 010070 016700 172574      MOV      TSTDAT,R0      ;FOR ERROR MSG
1637 010074 004767 003440      JSR      PC,ERROR       ;*ERROR* REPORT ERROR MESSAGE
1638 010100 000061              61             ;*****ERROR NUMBER 61*****
1639
1640 010102 062701 000002      60$:     ADD      #2,R1       ;UPPER WORD
1641 010106 026711 172560      CMP      TSTDAT+2,(R1)  ;THIS SHOULD BE UNCHANGED ALSO
1642 010112 001405              BEQ      65$             ;
1643 010114 016700 172552      MOV      TSTDAT+2,R0    ;
1644 010120 004767 003414      JSR      PC,ERROR       ;*ERROR* REPORT ERROR MESSAGE
1645 010124 000062              62             ;*****ERROR NUMBER 62*****
1646
1647 010126 105767 002720      65$:     TSTB     UPPFLG      ;WHICH LOOP ?
1648 010132 001003              BNE     66$             ;SECOND, BR OUT
1649 010134 105267 002712      INCB    UPPFLG         ;FIRST, KEEP GOING
1650 010140 000742              BR      55$             ;
1651 010142 005737 000406      66$:     TST      @#$PASS   ;FIRST PASS ?
1652 010146 001426              BEQ     85$             ;BR IF YES
1653 010150 005767 172526      70$:     TST      DBEMSK+2 ;LAST BIT ?
1654 010154 100404              BMI     75$             ;MINUS = BIT 31
1655 010156 004567 003172      JSR     R5,DASHL       ;SHIFT ROUTINE
1656 010162 002700              .WORD   DBEMSK         ;THIS 32 BIT WORD GETS SHIFTED
1657 010164 000643              BR      35$             ;
1658 010166 005767 172504      75$:     TST      SBEMSK+2  ;LAST BIT IN THIS MASK ?
1659 010172 100405              BMI     80$             ;BR IF LAST BIT
1660 010174 004567 003154      JSR     R5,DASHL       ;
1661 010200 002674              .WORD   SBEMSK         ;
1662 010202 000167 177454      JMP     T12B           ;
1663 010206 105737 000302      80$:     TSTB     @#PASFLG  ;FIRST PASS ?
1664 010212 001004              BNE     85$             ;BR IF SECOND
1665 010214 105237 000302      INCB    @#PASFLG       ;INDICATE SECOND PASS COMING
1666 010220 000167 177414      JMP     T12A           ;
1667 010224 016777 172460      172476 85$:  MOV     ECCDIS,@CSRADR ;
1668 010232 013701 000320      MOV     @#MINMEM,R1    ;RESTORE TEST ADDRESS
1669 010236 005021              CLR     (R1)+          ;CLEAR ANY DBE'S FROM TEST
1670 010240 005011              CLR     (R1)           ;
1671 010242 005077 172462      CLR     @CSRADR        ;CLEAR CSR
1672 010246 004767 003066      JSR     PC,RSTOT4      ;RESTORE REGS R0 TO R4
1673 010252 000167 172330      END12:  JMP     TSTSCP        ;TEST DUAL ADDRESS TEST
1674
1675
1676
1677
1678
1679 010256 122737 000013 000404 TST13:  CMPB    #13,@#$TESTN   ;CHECKS FOR DUAL ADDRESSING BY WRITING
1680 010264 001403              BEQ     6$              ;AND READING THE ADDRESS IN THE LOCATION
1681 010266 004767 003754      JSR     PC,SEQERR      ;AND WRITING AND READING ITS COMPLEMENT
1682 010272 000063              63             ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
1683
1684
1685 010274 005003              6$:     CLR     R3         ;R3 INDICATES FIRST PASS OR
1686 010276 010100              1$:     MOV     R1,R0     ;COMPLEMENT PASS
1687 010300 005703              TST     R3             ;IF R3= ZERO, STORE ADDRESS
1688 010302 001401              BEQ     2$             ;IN THE LOCATION
1689 010304 005100              COM     R0             ;OTHERWISE STORE COMPLEMENT

```

```

1690 010306 010021      2$:  MOV    R0,(R1)+      ;OF ADDRESS
1691 010310 020105      CMP    R1,R5          ;UNTIL HIGHEST LOC IS REACHED
1692 010312 103771      BLO   1$
1693 010314 020041      3$:  CMP    R0,-(R1)     ;CHECK FOR CORRECT CONTENTS
1694 010316 001405      BEQ   4$              ;BRANCH IF OK
1695 010320 105237 000275 INCB  @#$ADERR        ;PROBABLY AN ADDRESSING PROB.
1696 010324 004767 003210 JSR   PC,ERROR        ;*ERROR* REPORT ERROR MESSAGE
1697 010330 000064      64   ;*****ERROR NUMBER 64*****
1698
1699
1700 010332 010100      4$:  MOV    R1,R0
1701 010334 162700 000002  SUB   #2,R0          ;CHECK THAT ADDRESS IS STORED AT
1702 010340 005703      TST   R3              ;LOCATION IF R3 IS 0
1703 010342 001401      BEQ   5$              ;OTHERWISE CHECK FOR
1704 010344 005100      COM   R0              ;ADDRESS COMPLEMENT.
1705 010346 020104      5$:  CMP    R1,R4
1706 010350 101361      BHI   3$
1707 010352 112737 000001 000302 MOVB  #1,@#PASFLG     ;SET PASFLG FOR ERROR REPORT
1708 010360 005103      COM   R3              ;COMPLEMENT R3
1709 010362 001345      BNE   1$              ;REPEAT TEST, COMPLEMENTING ADE
1710 010364 000167 172216  END13: JMP   TSTSCP
1711
1712
1713      ;*(1) THIS TEST CHECKS THE MEMORY FOR THE CAPABILITY
1714      ;* OF HOLDING 1'S AND 0'S BY WRITING A BACKGROUND
1715      ;* OF BAKPAT AND READING IT
1716      ;*(2) MEMORY IS WRITTEN USING A BYTE AT A TIME
1717      ;*(3) STEPS 1 & 2 ARE REPEATED WITH A SWAPPED BACKGROUND PATTERN
1718
1719 010370 122737 000014 000404 TST14: CMPB  #14,@#$TESTN   ;CHECK FOR PROPER TEST SEQUENCE
1720 010376 001403      BEQ   1$
1721 010400 004767 003642      JSR   PC,SEQERR      ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
1722 010404 000065      65   ;*****ERROR NUMBER 65*****
1723
1724 010406 013700 000312      1$:  MOV    @#BAKPAT,R0   ;GET THE PATTERN INTO R0
1725 010412 110021      MOVB  R0,(R1)+        ;WRITE A BYTE
1726 010414 113721 000313      MOVB  @#BAKPAT+1,(R1)+ ;WRITE THE MEMORY WITH THE WORD STORED IN BAKPAT
1727 010420 020105      CMP   R1,R5          ;COMPARE TEST LOC TO TOP + 2
1728 010422 103771      BLO   1$              ;BRANCH IF LOWER
1729
1730 010424 020041      2$:  CMP    R0,-(R1)     ;TEST THE MEMORY TO SEE IF IT CONTAINS
1731      ;THE WORD STORED IN BAKPAT
1732 010426 001416      BEQ   8$
1733 010430 062701 000002  ADD   #2,R1
1734 010434 123741 000313      CMPB  @#BAKPAT+1,-(R1);CHECK FOR BYTE SELECTION PROBLEM
1735 010440 001402      BEQ   4$
1736 010442 120041      CMPB  R0,-(R1)     ;AGAIN CHECK FOR BYTE SELECTION PROBLEM
1737 010444 001002      BNE   6$
1738 010446 105237 000275  4$:  INCB  @#$ADERR        ;PREPARE TO INFORM THAT IT IS ADDRESSING ERROR
1739 010452 042701 000001  6$:  BIC   #1,R1          ;MAKE THE ADDRESS IN R1 EVEN
1740 010456 004767 003056      JSR   PC,ERROR        ;*ERROR* REPORT ERROR MESSAGE
1741 010462 000066      66   ;*****ERROR NUMBER 66*****
1742
1743 010464 020104      8$:  CMP    R1,R4
1744 010466 101356      BHI   2$
1745 010470 000337 000312      SWAB  @#BAKPAT        ;CHANGE THE DATA PATTERN

```



```

1746 010474 001744          BEQ      1$          ;IF THE DATA PATTERN DOES NOT HAVE LOW
1747                                     ; BYTE =0 THEN FALL THRU
1748 010476 000167 172104    END14: JMP      TSTSCP
1749
1750
1751                                     ;*THIS TEST IS CONCERNED ONLY WITH THE INTEGRITY
1752                                     ;*OF THE 4K MOS RAMS THAT STORE THE CHECKBITS.
1753                                     ;*USING DIAGA (BIT 2) AND BIT 13 TO PROTECT THE
1754                                     ;*PROGRAM IF NECESSARY, THE CHECKBITS ARE WRITTEN
1755                                     ;*AND READ VIA THE CSR.
1756
1757                                     :
1758 010502 122737 000015 000404 TST15: CMPB     #15,@#$TESTN ;CHECK FOR PROPER TEST SEQUENCE
1759 010510 001403          BEQ      1$
1760 010512 004767 003022    JSR      PC,ERROR ;*ERROR* REPORT ERROR MESSAGE
1761 010516 000067          BEQ      67          ;*****ERROR NUMBER 67*****
1762
1763 010520 010446          1$:      MOV      R4,-(SP) ;SAVE R4
1764 010522 016703 172164    MOV      DIAGA,R3 ;DIAGNOSTIC CHECK MODE
1765 010526 052703 004000    BIS      #4000,R3 ;CHECK BIT CT
1766 010532 013701 000320    MOV      @#MINMEM,R1 ;FIRST TEST LOC
1767 010536 010146          MOV      R1,-(SP) ;SAVE IT FOR LATER
1768 010540 005004          CLR      R4
1769 010542 010377 172162    MOV      R3,@CSRADR ;SET UP CSR
1770 010546 010411          2$:      MOV      R4,(R1) ;WRITE CHECKBITS BY WRITING LOC
1771 010550 012761 100000 000002 MOV      #100000,2(R1) ;SET BIT 31 TO MATCH CHECKBITS
1772 010556 062701 000004    ADD      #4,R1 ;POINT TO NEXT WORD
1773 010562 020105          CMP      R1,R5 ;TOP + 2
1774 010564 103770          BLO     2$ ;BR IF NOT
1775 010566 162701 000004    3$:      SUB      #4,R1 ;ADJUST R1
1776 010572 005711          TST     (R1) ;READ
1777 010574 032777 003740 172126 BIT      #3740,@CSRADR ;SHOULD BE ALL ZEROS
1778 010602 001421          BEQ     4$ ;BR IF OK
1779 010604 012700 004000    MOV      #4000,R0 ;GOOD DATA
1780 010610 004767 005040    JSR      PC,BITCHK ;USE XOR TO COUNT BAD CHECKBITS
1781 010614 032777 004000 172110 BIT      #4000,@SWR ;ECC DIS ?
1782 010622 001004          BNE     33$ ;PRNIT SINGLE ERRORS
1783 010624 022767 000001 005104 CMP      #1,BITCNT ;MORE THAN 1 BAD BIT ?
1784 010632 001405          BEQ     4$ ;YES
1785 010634 012700 004000    33$:     MOV      #4000,R0 ;FOR ERROR MSG
1786 010640 004767 002674    JSR      PC,ERROR ;*ERROR* REPORT ERROR MESSAGE
1787 010644 000070          BEQ     70          ;*****ERROR NUMBER 70*****
1788
1789 010646 052777 003740 172054 4$:      BIS      #3740,@CSRADR ;MAKE CHECKBITS ALL 1'S EXCEPT FOR CT
1790 010654 042777 004000 172046 BIC      #4000,@CSRADR ;MAKE CT = 0
1791 010662 010411          MOV      R4,(R1) ;WRITE THE CHECKBITS
1792 010664 010461 000002    MOV      R4,2(R1) ;BOTH WORDS
1793 010670 020116          CMP      R1,(SP) ;BOTTOM YET ?
1794 010672 001335          BNE     3$ ;BR IF NO
1795 010674 010377 172030    5$:      MOV      R3,@CSRADR ;RESTORE CSR
1796 010700 005711          TST     (R1) ;READ THE LOC AGAIN
1797 010702 017700 172022    MOV      @CSRADR,R0 ;GET CSR CONTENTS INTO R0
1798 010706 042700 170037    BIC      #170037,R0 ;ONLY WANT BITS 11-5
1799 010712 022700 003740    CMP      #3740,R0 ;READ FOR ALL 1'S
1800 010716 001421          BEQ     6$ ;BR IF OK
1801 010720 012700 003740    MOV      #3740,R0 ;FOR ERROR MSG

```

1802	010724	004767	004724			JSR	PC,BITCHK	:
1803	010730	032777	004000	171774		BIT	#4000,@SWR	:ECC DIS ?
1804	010736	001004				BNE	55\$	
1805	010740	022767	000001	004770		CMP	#1,BITCNT	:MORE THAN ONE ?
1806	010746	001405				BEQ	6\$:NO
1807	010750	012700	003740		55\$:	MOV	#3740,R0	:FOR MSG
1808	010754	004767	002560			JSR	PC,ERROR	:*ERROR* REPORT ERROR MESSAGE
1809	010760	000071				71		:*****ERROR NUMBER 71*****
1810								
1811	010762	010377	171742		6\$:	MOV	R3,@CSRADR	:RESTORE CSR
1812	010766	010411				MOV	R4,(R1)	:ACCESS THE LOC
1813	010770	012761	100000	000002		MOV	#100000,2(R1)	:SECOND WORD
1814	010776	062701	000004			ADD	#4,R1	:POINT TO NEXT ADDRESS
1815	011002	020105				CMP	R1,R5	:TOP + 2 YET?
1816	011004	103733				BLO	5\$:BR IF NOT
1817	011006	011601				MOV	(SP),R1	:RESTORE MINMEM
1818	011010	010377	171714		7\$:	MOV	R3,@CSRADR	:RESTORE CSR
1819	011014	005711				TST	(R1)	:READ
1820	011016	032777	003740	171704		BIT	#3740,@CSRADR	:SHOULD BR 0
1821	011024	001421				BEQ	8\$	
1822	011026	012700	004000			MOV	#4000,R0	:GOOD DATA
1823	011032	004767	004616			JSR	PC,BITCHK	:CHECK # OF BAD BITS
1824	011036	032777	004000	171666		BIT	#4000,@SWR	:ECC DIS ?
1825	011044	001004				BNE	77\$:YES
1826	011046	022767	000001	004662		CMP	#1,BITCNT	:MORE THAN ONE
1827	011054	001405				BEQ	8\$:NO
1828	011056	012700	004000		77\$:	MOV	#4000,R0	:FOR PRINTOUT
1829	011062	004767	002452			JSR	PC,ERROR	:*ERROR* REPORT ERROR MESSAGE
1830	011066	000072				72		:*****ERROR NUMBER 72*****
1831								
1832	011070	052777	003740	171632	8\$:	BIS	#3740,@CSRADR	:COMP CHECKBITS
1833	011076	042777	004000	171624		BIC	#4000,@CSRADR	:CLEAR CT
1834	011104	010411				MOV	R4,(R1)	:WRITE
1835	011106	010461	000002			MOV	R4,2(R1)	:
1836	011112	062701	000004			ADD	#4,R1	:NEXT WORD
1837	011116	020105				CMP	R1,R5	:TOP + 2 ?
1838	011120	103733				BLO	7\$:BR IF NO
1839	011122	010377	171602			MOV	R3,@CSRADR	:RESTORE CSR
1840	011126	162701	000004		9\$:	SUB	#4,R1	:ADJUST R1
1841	011132	005711				TST	(R1)	:READ
1842	011134	017700	171570			MOV	@CSRADR,R0	:GET CSR CONTENTS
1843	011140	042700	170037			BIC	#170037,R0	:ONLY WANT CHECKBITS
1844	011144	022700	003740			CMP	#3740,R0	:ALL ONES ?
1845	011150	001421				BEQ	10\$:BR IF OK
1846	011152	012700	003740			MOV	#3740,R0	:FOR ERROR MSG
1847	011156	004767	004472			JSR	PC,BITCHK	:COUNT BAD BITS
1848	011162	032777	004000	171542		BIT	#4000,@SWR	:ECC DIS
1849	011170	001004				BNE	99\$:YES
1850	011172	022767	000001	004536		CMP	#1,BITCNT	:MORE THAN ONE ?
1851	011200	001405				BEQ	10\$:NO
1852	011202	012700	003740		99\$:	MOV	#3740,R0	:FOR MSG.
1853	011206	004767	002326			JSR	PC,ERROR	:*ERROR* REPORT ERROR MESSAGE
1854	011212	000073				73		:*****ERROR NUMBER 73*****
1855								
1856	011214	010377	171510		10\$:	MOV	R3,@CSRADR	:RESTORE CSR
1857	011220	010411				MOV	R4,(R1)	:WRITE NEW CHECKBITS


```

1858 011222 012761 100000 000002      MOV    #100000,2(R1)    ;BIT 31
1859 011230 020116                CMP    R1,(SP)         ;BOTTOM YET ?
1860 011232 001335                BNE   9$              ;BR IF NOT
1861 011234 005711                TST   (R1)            ;RERAD
1862 011236 032777 003740 171464 11$:  BIT    #3740,@CSRADR  ;SHOULD BE ALL 0'S
1863 011244 001421                BEQ   12$             ;BR IF OK
1864 011246 012700 004000                MOV   #4000,R0        ;GOOD DATA
1865 011252 004767 004376                JSR   PC,BITCHK       ;COUNT BAD BITS
1866 011256 032777 004000 171446 11$:  BIT    #4000,@SWR      ;ECC DIS ?
1867 011264 001004                BNE   111$           ;YES
1868 011266 022767 000001 004442 111$:  CMP    #1,BITCNT      ;MORE THAN 1
1869 011274 001405                BEQ   12$             ;BR NO
1870 011276 012700 004000 111$:  MOV   #4000,R0        ;FOR MSG
1871 011302 004767 002232                JSR   PC,ERROR        ;*ERROR* REPORT ERROR MESSAGE
1872 011306 000074                74                   ;*****ERROR NUMBER 74*****
1873
1874 011310 062701 000004 12$:  ADD   #4,R1           ;POINT TO NEXT WORD
1875 011314 010377 171410                MOV   R3,@CSRADR      ;RESTORE CSR
1876 011320 020105                CMP   R1,R5           ;TOP + 2 YET?
1877 011322 103744                BLO   11$            ;BR IF NOT
1878 011324 005726                TST   (SP)+           ;FINALLY RESTORE STACK
1879 011326 012604                MOV   (SP)+,R4        ;RESTORE R4
1880 011330 000167 171252 111$:  JMP   TSTSCP          ;AND EXIT
1881
1882
1883      ;*(1) THIS TEST WRITES A BACK GROUND OF THE WORD STORED
1884      ;* AT BAKPAT.
1885      ;*(2) READS EVERY LOCATION FOR CORRECT DATA, SWAPS BYTES
1886      ;* AT THE LOCATION AND PROCEEDS IN MAX. TO MIN
1887      ;* DIRECTION OF MEMORY LOCATIONS.
1888      ;*(3) READS EVERY LOCATION FOR SWAPPED BAKPAT PATTERN
1889      ;* WRITES BAKPAT BACKGROUND IN THE LOCATION AND PROCEEDS
1890      ;* IN MIN. TO MAX. DIRECTION
1891      ;*(4) REPEATS STEP 2 GOING IN MIN. TO MAX. DIRECTION
1892      ;*(5) REPEATS STEP 3 GOING IN MAX. TO MIN. DIRECTION
1893
1894
1895 011334 122737 000016 000404 TST16:  CMPB   #16,@#STESTN    ;CHECK FOR PROPER TEST SEQUENCE
1896 011342 001403                BEQ   1$              ;
1897 011344 004767 002676                JSR   PC,SEQERR       ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
1898 011350 000075                75                   ;*****ERROR NUMBER 75*****
1899
1900 011352 004737 000120 1$:  JSR   PC,@#WRTMEM     ;GO TO WRITE THE MEMORY WITH A BACKGROUND OF THE
1901                                ;WORD STORED IN BAKPAT
1902 011356 020041 2$:  CMP   R0,-(R1)        ;READ THE CONTENTS OF LOCATION POINTED BY R1
1903 011360 001403                BEQ   3$              ;TO SEE IF IT HAS THE SAME VALUE AS R0
1904 011362 004767 002152                JSR   PC,ERROR        ;*ERROR* REPORT ERROR MESSAGE
1905 011366 000076                76                   ;*****ERROR NUMBER 76*****
1906
1907 011370 000300 3$:  SWAB  R0              ;SWAP THE BYTES AT (R1)
1908 011372 010011                MOV   R0,(R1)         ;READ (R1) FOR CORRECT VALUE
1909 011374 021100                CMP   (R1),R0
1910 011376 001403                BEQ   4$              ;
1911 011400 004767 002134                JSR   PC,ERROR        ;*ERROR* REPORT ERROR MESSAGE
1912 011404 000077                77                   ;*****ERROR NUMBER 77*****
1913

```

```

1914
1915 011406 000300      4$:  SWAB  R0      ;SWAP THE BYTES OF THE REGISTER
1916                                     ;CONTAINING BACKGROUND PATTERN
1917 011410 001023      BNE  9$      ;IF THE LOWER BYTE OF THE REGISTER
1918                                     ;IS NOT 0 THEN THE PROGRAM IS READING
1919                                     ;THE MEMORY TO CONTAIN A BACK GROUND OF
1920
1921                                     ;BAKPAT AND WRITING THE SWAPPED WORD
1922
1923                                     ;IN WHICH CASE GO TO 9$
1924
1925 011412 005703      5$:  TST  R3      ;R3 WAS 0 WHEN THE PROGRAM ENTERED
1926                                     ;THIS TEST, AND IT IS NOT ALTERED UNTIL PASFLG=3
1927                                     ;IF R3 EQUAL 0 THEN THE PROGRAM IS
1928                                     ;READING/WRITING MIN. TO MAX. OTHERWISE
1929                                     ;IT IS GOING IN MAX. TO MIN. DIRECTION
1930 011414 001023      BNE  10$     ;IF R3 IS NOT CLEAR THEN GO TO 10$
1931 011416 062701 000002 6$:  ADD  #2,R1    ;OTHERWISE ADD 2 TO THE CONTENTS OF R1
1932 011422 020105      CMP  R1,R5     ;COMPARE R1 WITH THE MAX. MEMORY LOCATION TO
1933                                     ;BE TESTED
1934 011424 103006      BHIS  8$      ;IF R1>R5 THEN GO TO 8$ OTHERWISE
1935 011426 020011      CMP  R0,(R1)  ;READ (R1) FOR THE CORRECT DATA
1936 011430 001757      BEQ  3$      ;WRITE COMPLEMENT OF THE DATA FOUND AT (R1)
1937                                     ;AND REPEAT UNTIL R1 > R5
1938 011432 004767 002102 JSR  PC,ERROR ;*ERROR* REPORT ERROR MESSAGE
1939 011436 000100      100
1940
1941 011440 000753      BR   3$      ;*****ERROR NUMBER 100*****
1942 011442 105237 000302 8$:  INCB @#PASFLG
1943 011446 000300      SWAB R0
1944 011450 001742      BEQ  2$      ;IF THE LOWER BYTE OF R0 IS ALL 0'S
1945                                     ;THEN BEGIN READING BAKPAT SWAPPED WRITING BAKPAT
1946                                     ;AND READING BAKPAT GOING FROM MAX. TO MIN.[PASFLG=4]
1947 011452 005103      COM  R3
1948 011454 010401      MOV  R4,R1
1949 011456 000763      BR   7$      ;OTHERWISE CLEAR R0
1950                                     ;PUT THE LOWEST TESTING ADDRESS IN R1
1951                                     ;AND BEGIN READING 0'S, WRITING 1'S AND
1952                                     ;READING 1'S IN MIN. TO MAX. DIRECTION [PASFLG=3]
1953 011460 005703      9$:  TST  R3
1954 011462 001353      BNE  5$
1955                                     ;IF R3 IS NON 0, I.E. PASFLG=3
1956                                     ;THEN READ BAKPAT, WRITE
1957                                     ;SWAPPED BAKPAT AND READ SWAPPED BAKPAT
1958 011464 020104      10$: CMP  R1,R4  ;IN MIN. TO MAX. DIRECTION
1959                                     ;OTHERWISE TEST IS PROCEEDING IN MAX. TO
1960 011466 101333      BHI  2$      ;MIN. DIRECTION.
1961 011470 105237 000302 INCB @#PASFLG ;KEEP ON LOOPING UNTIL R1=R4
1962 011474 000300      SWAB R0
1963 011476 001753      BEQ  7$
1964 011500 000167 171102 END16: JMP  TSTSCP ;IF R0 SWAPPED HAS LOWER BYTE=0
1965                                     ;THEN READ BAKPAT SWAPPED, WRITE BAKPAT,
1966                                     ;AND READ BAKPAT GOING FROM MIN. TO MAX.
1967
1968 ;*(1) THIS TEST WRITES THE MEMORY WITH A BACK GROUND OF BAKPAT
1969 ;*(2) WITH PASFLG=0 THE TEST READS THE MEMORY FOR BAKPAT
; * AND THEN INCREMENTS PASFLG

```



```

1970      ;*(3)  IT THEN READS/SWAPS BYTES/WITES A LOCATION X FOR
1971      ;*      OVER 2 MSEC AND THEN READS THE MEMORY FOR BAKPAT
1972      ;*(4)  REPEATS STEP 3 WITH X=X+4K UNTIL END OF MEMORY IS ENCOUNTERED
1973      ;*(5)  IT THEN INCREMENTS PASFLG AND WRITES THE MEMORY TO
1974      ;*      BAKPAT AND WITH PASFLG=2 IT READS MEMORY FOR ALL
1975      ;*      SWAPPED BAKPAT AFTER WHICH PASFLG IS INCREMENTED TO 3
1976      ;*(6)  REPEATS STEPS 3 AND 4 READING THE MEMORY FOR SWAPPED
1977      ;*      BAKPAT INSTEAD OF BAKPAT.
1978 011504 122737 000017 000404 TST17: CMPB   #17,@#STESTN ;CHECK FOR PROPER TEST SEQUENCE
1979 011512 001403          BEQ     .+10
1980 011514 004767 002526          JSR     PC,SEQERR ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
1981 011520 000101          ;*****ERROR NUMBER 101*****
1982
1983 011522 004737 000120          RPT17: JSR     PC,@#WRTMEM ;GO TO WRITE THE MEMORY WITH A BACKGROUND OF THE
1984          ;WORD STORED AT LOCATION BAKPAT
1985 011526 005037 000302          CLR     @#PASFLG
1986 011532 010403          1$:    MOV     R4,R3 ;SET R3
1987 011534 010401          2$:    MOV     R4,R1 ;AND R1 TO THE STARTING ADDRESS OF MEMORY UNDER TEST
1988 011536 020011          3$:    CMP     R0,(R1) ;CHECK (R1) FOR CORRECT DATA
1989 011540 001403          BEQ     4$
1990 011542 004767 001772          JSR     PC,ERROR ;*ERROR* REPORT ERROR MESSAGE
1991 011546 000102          102    ;*****ERROR NUMBER 102*****
1992
1993 011550 062701 000002          4$:    ADD     #2,R1 ;INCREMENT R1 BY 2
1994 011554 020105          CMP     R1,R5 ;SEE IF R1 HAS REACHED THE MAX. OF MEMORY
1995 011556 103767          BLO     3$
1996 011560 132737 000001 000302 BITB   #1,@#PASFLG ;CHECK TO SEE IF PASFLG=0 OR 2
1997 011566 001002          BNE     5$
1998 011570 105237 000302          INCB   @#PASFLG ;IN WHICH CASE INCREMENT PASFLG COUNTER BY 1
1999
2000 011574 020305          5$:    CMP     R3,R5 ;SEE IF R3 HAS REACHED THE MAX. OF THE MEMORY
2001 011576 103012          BHIS   7$
2002 011600 012702 037776          MOV     #37776,R2 ;WRITE INTO 1 LOC FOR >2MS (ABOUT 100MS)
2003
2004 011604 000313          6$:    SWAB   (R3)
2005 011606 005302          DEC     R2
2006 011610 001375          BNE     6$
2007 011612 010337 000350          MOV     R3,@#SAVLOC ;SAVE LOCATION WRITTEN FOR 2MS FOR ERROR REPORT.
2008 011616 062703 020000          ADD     #20000,R3 ;BY ADDING 1 TO THE 14TH ADDRESS BIT CAUSE
2009          ;R3 TO POINT TO A LOCATION IN THE NEXT
2010          ;4K BANK OF MEMORY
2011 011622 000744          BR     2$
2012 011624 105237 000302          7$:    INCB   @#PASFLG ;MAKE PASFLG=2
2013 011630 000337 000312          SWAB   @#BAKPAT ;IF BAKPAT IS NOT BEING SWAPPED FOR THE 2ND
2014 011634 001732          BEQ     RPT17 ;THEN GO BACK TO THE LOCATION RPT17
2015 011636 000167 170744          END17: JMP     TSTSCP
2016
2017
2018
2019
2020
2021
2022
2023

```

```
2024 011642 005077 171062 RELOC: CLR @CSRADR ;
2025 011646 012737 000377 000312 MOV #377,@#BAKPAT ;
2026 011654 105737 000272 TSTB @#MMAVA ;IS THE MEMORY MANAGEMENT BEING TESTED ?
2027 011660 001170 BNE CONTMM ;IF SO THEN GO TO CONTMM AND CONTINUE TESTING
2028 ;MEMORY MANAGEMENT
2029 011662 005767 171030 TST LDFLG ;ONLY RELOCATE IF IN MEM UNDER TEST
2030 011666 001542 BEQ CKDONE ;DON'T RELOC, BRANCH
2031 011670 032777 001000 171034 BIT #1000,@SWR ;RELOCATION WANTED?
2032 011676 001136 BNE CKDONE ;BRANCH IF NO
2033 011700 005767 171022 TST INHREL ;RELOCATION ALLOWED ?
2034 011704 001133 BNE CKDONE ;NO, SINGLE ERROR IN LOC 0-500, BRANCH
2035 011706 105737 000170 TSTB @#REL ;IF THE PROGRAM HAS ALREADY BEEN RELOCATED THEN ALSO
2036 011712 100473 BMI RELOER ; PLACE THE PROGRAM BACK IN LOWER CORE
2037 011714 112737 000200 000170 MOVB #200,@#REL ;OTHERWISE PREPARE TO RELOCATE
2038
2039 ;RELOCATE THE DIAGNOSTIC TO SECOND 16K
2040
2041
2042 011722 012701 100500 MOV #100000+BEGIN,R1;START OF RELOCATED VERSION
2043 011726 012702 116154 MOV #100000+ENDPROG,R2;END
2044 011732 005711 3$: TST (R1) ;READ A WORD
2045 011734 032777 100000 170766 BIT #100000,@CSRADR ;CHECK FOR DBE
2046 011742 001005 BNE 5$ ;BR IF ERROR
2047 011744 062701 000004 ADD #4,R1 ;NEXT WORD
2048 011750 020102 CMP R1,R2 ;END YET ?
2049 011752 103767 BLO 3$ ;BR IF NOT
2050 011754 000405 BR 4$ ;BR IF ALL IS OK
2051 011756 105037 000170 5$: CLRB @#REL ;INDICATE NO RELOC
2052 011762 004767 002260 JSR PC,FATERR ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
2053 011766 000103 103 ;*****ERROR NUMBER 103*****
2054
2055 011770 032777 000040 170734 4$: BIT #40,@SWR ;PRINTING WANTED
2056 011776 001005 BNE 1$ ;BR IF NO
2057 012000 004767 002466 JSR PC,PNTMES ;TYPE 'RELOC'
2058 012004 042522 047514 000103 .ASCIZ /RELOC/
2059 .EVEN
2060 012012 004767 003272 1$: JSR PC,CLRMM ;CLEAR MM REGS
2061 012016 012737 000500 000320 MOV #BEGIN,@#MINMEM ;NEW TEST ADDRESS
2062 012024 052767 000010 170656 BIS #10,ECCDIS ;PROTECT 2ND 16K
2063 012032 052767 000010 170652 BIS #10,DIAGA ;DITTO
2064 012040 012705 100000 MOV #100000,R5 ;START OF 2ND 16K
2065 012044 010567 170650 MOV R5,BASE ;BASE FOR RELOC
2066 012050 012704 000430 MOV #BEGIN-50,R4 ;FIRST LOC TO BE RELOCATED
2067 012054 060405 ADD R4,R5 ;
2068 012056 012425 2$: MOV (R4)+,(R5)+ ;MOV A WORD
2069 012060 020437 000342 CMP R4,@#SAVR4 ;LAST LOC YET ?
2070 012064 103774 BLO 2$ ;BR IF NO
2071 012066 012704 000430 MOV #BEGIN-50,R4 ;FIRST TEST ADDRESS
2072 012072 016705 170622 MOV BASE,R5
2073 012076 000137 100500 JMP @#BEGIN+100000 ;JUMP TO RELOCATED VERSION
2074
2075 ;*RELOCATE THE DIAGNOSTIC BACK TO LOWER MEMORY
2076
2077
2078 012102 105737 000170 RELOER: TSTB @#REL ;IS DIAGNOSTIC IN RELOCATED STATE?
2079 012106 100032 BPL CKDONE ;BRANCH IF NO
```



```

2080 012110 005067 170604          CLR    BASE
2081 012114 042767 000010 170566    BIC    #10,ECCDIS
2082 012122 042767 000010 170562    BIC    #10,DIAGA
2083
2084 012130 016737 170566 000320    MOV    SAVMIN,@#MINMEM ;RESTORE TEST ADDRESS FOR ECC
2085 012136 012704 000430          MOV    #BEGIN-50,R4 ;PREPARE TO RELOCATE THE PROGRAM TO LOWER CORE
2086 012142 010405          MOV    R4,R5 ;
2087 012144 062705 100000          ADD    #100000,R5 ;
2088 012150 012524          2$:   MOV    (R5)+,(R4)+
2089 012152 020437 000342          CMP    R4,@#SAVR4
2090 012156 103774          BLO   2$
2091 012160 105037 000170          CLRB  @#REL
2092 012164 012706 000500          MOV    #BEGIN,SP ;RESET STACK TO LOWER MEMORY
2093 012170 010637 000346          MOV    SP,@#SAVR6 ;'BEGIN' USES THIS TO RESET THE STACK.
2094 012174 000137 012200          CKDONE: JMP   @#LOWER ;TRANSFER THE PROGRAM CONTROL TO THE LOWER CORE
2095
2096
2097
2098 012200 105737 000402          LOWER: TSTB  @#SFATAL ;FATAL ERROR
2099 012204 001405          BEQ   1$ ;BR IF NOT
2100 012206 005737 000042          TST  @#42 ;APT ?
2101 012212 001402          BEQ   1$ ;NO KEEP TESTING
2102 012214 000167 002072          JMP   APTHLT
2103 012220 105737 000311          1$:   TSTB  @#SAVKBB ;HERE DUE TO ^C TYPED?
2104 012224 001141          BNE   $TPSTK ;BRANCH IF YES (TYPE ERROR STACK)
2105 012226 004767 002506          TSTMM: JSR   PC,MEMMNG ; SET THE REGISTERS IF THE MEMORY MANAGEMENT
2106 ;IS AVAILABLE
2107 012232 105737 000272          TSTB  @#MMAVA ;IS MEM. MANAG. AVAILABLE ?
2108 012236 001474          BEQ   ENDPAS ;BRANCH IF NO
2109 012240 000406          BR    $CNTMM ;BEGIN TESTING ABOVE 28K
2110 012242 022737 007400 172352  CONTMM: CMP   #7400,@#172352 ;DON'T WANT TO WRAP AROUND
2111 012250 001464          BEQ   ENDMAX ;GO TO END-OF-PASS
2112 012252 004767 002640          JSR   PC,UPMM ;GO TO UPDATE MEM. MANAG. REGISTERS
2113 012256 012703 000322          $CNTMM: MOV  #LOWTWO,R3 ;MAKE R3 POINT TO THE LOCATION LOWTWO
2114 012262 004767 002746          JSR   PC,GETSIZ ; LOAD BITS 6-10 OF R2 WITH THE BITS 13-17
2115 ;OF THE LOWEST ADDRESS UNDER TEST
2116 012266 012704 040000          MOV   #40000,R4 ;MAKE R4 POINT TO THE LOWEST LOCATION IN THE BANK
2117 ;POINTED BY PAGE ADDRESS REGISTER 2 (PAR2)
2118 012272 020237 172344          CMP   R2,@#172344 ;IS THE CONTENT OF R2 LOWER THAN THE CONTENT OF
2119 ;PAR2 ?
2120 012276 103405          BLO   2$ ;IF SO THEN GO 2$
2121 012300 050104          BIS   R1,R4 ;SUBROUTINE GETSIZ LOADED R1 WITH BITS 0-12
2122 ;OF LOWADD WHICH HAVE NOW BEEN LOADED IN R4
2123 012302 162702 000200          SUB   #200,R2
2124 012306 004767 002432          JSR   PC,MMREG ; SET MEM. MANAG. REGISTERS
2125 012312 010437 000320          2$:   MOV   R4,@#MINMEM ;NEW MINMEM FOR ECC TESTS
2126 012316 004767 002712          JSR   PC,GETSIZ ;PLACE BITS 13-17 OF HIGHEST LOCATION TO BE TESTED
2127 ;IN BITS 6-10 OF R2, #160000 IN R0 AND BITS 0-12
2128 ;OF LOCATION HIGHADD IN R1
2129 012322 004767 000020          JSR   PC,MAXADR ; GET THE ADDRESS OF MAX. MEM. UNDER TEST
2130 012326 010005          MOV   R0,R5
2131 012330 004767 002700          JSR   PC,GETSIZ ; PREPARE TO SET UP LOCATION MAXMEM
2132 012334 004767 000006          JSR   PC,MAXADR ; GET THE MAXIMUM ADDRESS OF AVAILABLE MEMORY
2133 012340 010013          MOV   R0,(R3) ;AND STORE INTO 'MAXMEM'
2134 012342 000167 167664          JMP   CLRMEM ;GO TEST A 20K SLICE ABOVE 28K.
2135

```

2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148

012346 010046
012350 012700 172356
012354 162716 020000
012360 050116
012362 020240

:MAXADR - SUBROUTINE TO GET CURRENT 20K SLICE OF MEMORY ADDRESSES ABOVE 28K.
:REGISTERS:
:R0= ON ENTRY= #160000 AND ON EXIT=HIGHEST VIRTUAL ADDR. UNDER TEST
:R1= LOW ORDER 12 BITS OF VIRTUAL TEST ADDRESS
:R2= PAR BLOCK NO. CURRENTLY UNDER TEST.
MAXADR: MOV R0, -(SP) ;PUT MAXIMUM AVAILABLE ADDRESS ON STACK
MOV #172356, R0 ;R0=PAR7 UNIBUS ADDRESS
:**BEGIN LOOP TO FIND PAR ADDRESS UNDER TEST
2\$: SUB #20000, (SP) ;DECREMENT VIRTUAL ADDRESS BY 4K
BIS R1, (SP) ;SET BITS 11:0 TO MAXIMUM VIRTUAL TEST ADDRESS
CMP R2, -(R0) ;DOES CURRENT PAR= TEST BLOCK NO.?


```

2149 012364 001411          BEQ      3$          ;BRANCH IF YES
2150 012366 020027 172340    CMP      R0,#172340 ;ARE WE AT PAR0?
2151 012372 101370          BHI      2$          ;NO KEEP TRYING
2152                                ;**END LOOP TO FIND PAR ADDRESS UNDER TEST
2153 012374 062700 000004    ADD      #4,R0      ;SET TO CURRENT PAR
2154 012400 021002          CMP      (R0),R2    ;IS THE PAR BLOCK UNDER TEST GTR THAN ALLOWED?
2155 012402 003006          BGT      4$          ;BRANCH IF YES (FALL INTO ENDPAS)
2156 012404 012716 157776    MOV      #157776,(SP) ;EXIT WITH MAXADR= 28K SEGMENT TEST SIZE
2157 012410 012600          3$: MOV      (SP)+,R0 ;SET R0 TO MAXIMUM VIRTUAL TEST ADDRESS
2158 012412 062700 000002    ADD      #2,R0      ;MAKE MAXIMUM MEMORY+2
2159 012416 000207          RTS      PC         ;AND EXIT MAXADR ROUTINE
2160
2161 012420 022626          4$:  CMP      (SP)+,(SP)+ ;FIXUP STACK
2162                                ;AND FALL THRU TO ENDPAS.
2163 012422 016737 170274 000320  ENDMAX: MOV      SAVMIN,@#MINMEM ;BECAUSE WE WON'T SIZE AGAIN
2164                                ;* TYPE ROUTINE FOR ERROR STACK
2165                                ;* -----
2166                                ;*
2167                                ;*
2168                                ;* THIS ROUTINE IS USED TO DETERMINE IF TYPE OUT OF THE ERROR STACK
2169                                ;* FOR ONLY THE FAILING BITS IS REQUIRED OR NOT
2170
2171
2172 012430 026727 170274 172134  ENDPAS: CMP      CSRADR,#172134 ;SECOND CSR ?
2173 012436 001004          BNE      2$          ;BR IF NOT
2174 012440 012767 172136 170262  1$: MOV      #172136,CSRADR ;RESTORE INIT ADDRESS
2175 012446 000424          BR       4$          ;
2176 012450 012767 172134 170252  2$: MOV      #172134,CSRADR ;GET READY TO LOOK FOR SECOND CSR
2177 012456 012737 012512 000004    MOV      #3$,@#4    ;SET UP FOR TRAP
2178 012464 012777 020020 170236    MOV      #20020,@CSRADR ;MAKE SURE IT BELONGS TO A MS11K
2179 012472 032777 020020 170230    BIT      #20020,@CSRADR ;THESE BITS ARE UNIQUE TO MS11K CSR
2180 012500 001757          BEQ      1$          ;BR IF NOT MS11K CSR
2181 012502 005077 170222          CLR      @CSRADR    ;CLEAR THE BITS JUST IN CASE
2182 012506 000167 000222          JMP      ACT11      ;GUESS IT DOES...GO TEST IT
2183 012512 062706 000004          3$: ADD      #4,SP    ;RESTORE SP
2184 012516 000750          BR       1$          ;
2185 012520 032777 020000 170204  4$: BIT      #20000,@SWR ;ARE WE GOING TO TYPE THE ERROR STACK AT END OF PASS?
2186 012526 001051          BNE      $EOP       ;IF NOT THEN GO TO $EOP
2187 012530 012746 177777          $TPSTK: MOV     #-1,-(SP) ;THE PROGRAM HAS REACHED THE END AND ERROR
2188                                ;STACK AND END OF PASS WILL BE TYPED OUT
2189 012534 012701 016154          MOV      #ENDPROG,R1 ;PLACE THE STARTING ADDRESS OF THE ERROR STACK
2190                                ;FOR 0 TO 4K MEMORY IN R1
2191 012540 012703 000376          TYPSTK: MOV     #376,R3
2192 012544 005216          INC      (SP)       ;IF WE HAVE GONE THRU THE ENTIRE
2193 012546 020137 000304          CMP      R1,@#ENDSTK ;HAS THE END OF THE ERROR STACK BEEN REACHED ?
2194 012552 103037          BHIS    $EOP       ;THEN GO TO TYPE END OF PASS
2195 012554 112702 000022          MOVB   #18.,R2
2196 012560 105302          RETSTK: DECB   R2
2197 012562 002766          BLT     TYPSTK
2198
2199 012564 105721          TSTB   (R1)+
2200 012566 001774          BEQ    RETSTK
2201 012570 020227 000020          CMP    R2,#16.
2202
2203
2204 012574 103403          BLO    2$          ;IF NOT THEN GO TO TYPE BIT NUMBER

```

```

2205 012576 004767 001522          JSR    PC,TPADER      ;OTHERWISE TYPE 'ADDRESS ERROR'
2206 012602 000404                   BR     FAILNM
2207 012604 010237 000306          2$:   MOV    R2,@#DECWRD ;PREPARE TO TYPE THE NUMBER OF THE FAILING BIT
2208                                     ;IN DECIMAL
2209 012610 004767 001670          JSR    PC,TYPDEC      ;GO TO TYPE THE BIT NUMBER IN DECIMAL
2210 012614 011637 000306          FAILNM: MOV   (SP),@#DECWRD ;PREPARE TO TYPE THE PAGE NUMBER
2211 012620 004767 001664          JSR    PC,$TPDEC     ;IN DECIMAL
2212 012624 005043                   CLR    -(R3)
2213 012626 114113                   MOVB   -(R1),(R3)     ;PREPARE TO PRINTOUT THE NUMBER OF TIMES THIS
2214                                     ;FAILURE OCCURED
2215 012630 105021                   CLRB   (R1)+         ;CLEAR THE ERROR STACK
2216 012632 005043                   CLR    -(R3)
2217 012634 105237 000310          INCB   @#TYPCNT     ;ENABLE THE TYPE OUT OF 1 WORDS
2218 012640 004767 002004          JSR    PC,RPTOCT    ;TYPE THE 4K BANK AND THE NUMBER OF TIMES
2219                                     ;THIS FAILURE WAS SEEN
2220 012644 012703 000376          MOV    #376,R3      ;RESET SCRATCH STACK FOR EACH BIT PRINTED.
2221 012650 000743                   BR     RETSTK
2222
2223
2224                                     ;* END OF PASS
2225                                     ;* -----
2226
2227
2228                                     ;*
2229                                     ;* TYPE 'END PASS' AND DISABLE PARITY.
2230                                     ;* ALSO SERVICE ACT11.
2231                                     ;* AND EVERY CONSECUTIVE PASSES UNLESS BIT 4 OF $SWREG IS HIGH
2232                                     ;*
2233
2233 012652 005002                   $EOP:  CLR    R2      ;SET R2= PARITY MODULE DISABLE CODE
2234 012654 105737 000311          TSTB   @#SAVKBB     ;CONTROL-C TYPED?
2235 012660 001050                   BNE    CTLC         ;BRANCH IF YES-RESTORE LOADERS AND HALT-
2236 012662 005237 000406          INC    @#$PASS     ;INCREMENT PASS COUNT
2237 012666 005067 170032          CLR    ERRFLG      ;RESET ERROR HEADER FOR NEXT PASS
2238 012672 032777 000040 170032   BIT    #40,@SWR    ;'END PASS #XX' PRINTOUT WANTED?
2239 012700 001015                   BNE    ACT11       ;BRANCH IF NO
2240 012702 004767 001556          TYPEOP: JSR   PC,TPCRLF ;TYPE CR, LF, AND 'END PASS #'
2241 012706 047105 020104 040520   .ASCIZ /END PASS #/
2242 012714 051523 021440 000
2243
2244                                     .EVEN
2244 012722 013737 000406 000306   MOV    @#$PASS,@#DECWRD ;GET PASS COUNT
2245 012730 004767 001554                   JSR    PC,$TPDEC     ;TYPE IT
2246 012734 013700 000042          ACT11: MOV   @#42,R0   ;GET THE MONITOR ADDRESS
2247 012740 001405                   BEQ    $DOAGN       ;IF NONE
2248 012742 004767 000012          JSR    PC,RLODER    ;RESTORE XXDP MONITOR
2249 012746 000005                   RESET   ;RETURN TO ACT11 MONITOR.
2250
2251
2252                                     ;* SERVICE XXDP/ACT11
2253 012750 000137 000156          JMP    @#$ENDAD     ;JUMP TO ACT SERVICE
2254
2255 012754 000137 000250          $DOAGN: JMP   @#RESTRT ;REPEAT TEST IF NOT UNDER ACT11/XXDP
2256
2257 012760 004767 002324          RLODER: JSR   PC,CLRMM ;STOP MEMORY MANAGEMENT SO CAN RESTORE LOADERS
2258 012764 013704 000342          MOV    @#SAVR4,R4  ;RESTORE R4 WITH SAVR4
2259 012770 014445          4$:   MOV    -(R4),-(R5) ;RESTORE LOADERS
2260 012772 020437 000304          CMP    R4,@#ENDSTK

```



```

2261 012776 101374          BHI 4$
2262 013000 000207          RTS PC          ;RETURN FROM RLODER CALL
2263
2264          ;CONTROL C HANDLER
2265
2266 013002 004767 177752    CTLC: JSR PC,RLODER    ;RESTORE ABS LOADER
2267 013006 000167 001300    JMP APTHLT          ;IF NOT APT HALT AT FATHLT
2268
2269
2270          ;:*****
2271
2272          ;CHECK BIT GENERATOR ROUTINE
2273          ;CALL: JSR PC,CHKGEN
2274          ;
2275          ; SOURCE = ADDRESS OF DATA
2276          ; CHECK = WORD CONTAINING GENERATED CHECKBITS
2277
2278 013012 000000    SOURCE: .WORD 0
2279 013014 000000    CHECK: .WORD 0
2280 013016 125252    CHKTAB: .WORD 125252          ;C1 BIT TABLE
2281 013020 125252    .WORD 125252          ;
2282 013022 146314    .WORD 146314          ;C2 BIT TABLE
2283 013024 146314    .WORD 146314          ;
2284 013026 170360    .WORD 170360          ;C4 BIT TABLE
2285 013030 170360    .WORD 170360          ;
2286 013032 177777    .WORD 177777          ;C8 BIT TABLE
2287 013034 177400    .WORD 177400          ;
2288 013036 000377    .WORD 377          ;C16 BIT TABLE
2289 013040 177777    .WORD 177777          ;
2290 013042 177400    .WORD 177400          ;C32 BIT TABLE
2291 013044 177777    .WORD 177777          ;
2292 013046 064551    .WORD 64551          ;CT BIT TABLE
2293 013050 113151    .WORD 113151          ;
2294
2295 013052 000          UPPFLG: .BYTE 0
2296
2297 013053 000          CHKCNT: .BYTE 0          ;C1 PARITY COUNTER
2298 013054 000          .BYTE 0          ;C2 PARITY COUNTER
2299 013055 000          .BYTE 0          ;C4 PARITY COUNTER
2300 013056 000          .BYTE 0          ;C8 PARITY COUNTER
2301 013057 000          .BYTE 0          ;C16 PARITY COUNTER
2302 013060 000          .BYTE 0          ;C32 PARITY COUNTER
2303 013061 000          .BYTE 0          ;CT PARITY COUNTER
2304 013062 077          PARBYT: .BYTE 77          ;ODD/EVEN PARITY FLAGS
2305          ;0=EVEN
2306          ;1=ODD
2307          ;CT C32 C16 C8 C4 C2 C1
2308
2309 013064 004467 000236    CHKGEN: JSR R4,SAVOT4    ;SAVE R0 TO R4
2310 013070 005000          CLR R0          ;CLEARING OUT PARITY COUNTERS
2311 013072 012701 013053    MOV #CHKCNT,R1    ;START 7 COUNTERS
2312 013076 066701 167616    ADD BASE,R1
2313 013102 022700 000007    10$: CMP #7,R0
2314 013106 001403          BEQ 20$
2315 013110 105021          CLRB (R1)+
2316 013112 005200          INC R0          ;BUMP THE COUNT

```

2317	013114	000772		BR	10\$:LOOP
2318	013116	012701	000001	20\$: MOV	#1,R1		:SET THE TEST BIT
2319	013122	012702	013016	30\$: MOV	#CHKTAB,R2		:START OF CHECK BIT TABLE
2320	013126	066702	167566	ADD	BASE,R2		
2321	013132	012703	013053	MOV	#CHKCNT,R3		:START ADDRESS OF PARITY COUNTERS
2322	013136	066703	167556	ADD	BASE,R3		
2323	013142	012704	000006	MOV	#6,R4		:FOR SEVEN PARITY COUNTERS
2324	013146	105067	177700	40\$: CLRB	UPPFLG		:INDICATE LOWER WORD
2325	013152	016700	177634	MOV	SOURCE,R0		:SET ADDRESS OF DATA IN R0
2326	013156	066700	167536	ADD	BASE,R0		
2327	013162	030110		42\$: BIT	R1,(R0)		:CHECK LOWER 16 BITS FOR TEST BIT
2328	013164	001403		BEQ	50\$:BRANCH IF NOT SET
2329	013166	030112		BIT	R1,(R2)		:LOOK FOR TEST BIT IN CHECK BIT TABLE
2330	013170	001401		BEQ	50\$:BRANCH IF NOT FOUND
2331	013172	105213		INCB	(R3)		:FOUND A MATCH, COUNT IT
2332	013174	062700	000002	50\$: ADD	#2,R0		:R0 NOW = SOURCE +2
2333	013200	062702	000002	ADD	#2,R2		:POINT TO SECOND WORD IN TABLE
2334	013204	105767	177642	TSTB	UPPFLG		:CHECK FOR UPPER/LOWER
2335	013210	001003		BNE	60\$:BRANCH IF UPPER
2336	013212	105267	177634	INCB	UPPFLG		:CHANGE FLAG TO UPPER
2337	013216	000761		BR	42\$:CHECK SECOND 16 BITS
2338	013220	005704		60\$: TST	R4		:LOOK FOR LAST PARITY COUNTER
2339	013222	001403		BEQ	70\$:BRANCH IF DONE
2340	013224	005203		INC	R3		:ADVANCE PARITY COUNTER POINTER
2341	013226	005304		DEC	R4		:ADVANCE CHECK BIT FLAG COUNTER
2342	013230	000746		BR	40\$:GO CHECK NEXT CHECK BIT IN TABLE
2343	013232	006301		70\$: ASL	R1		:SHIFT THE BIT OVER
2344	013234	103332		BCC	30\$:ADJUST THE PARITY AND ASSEMBLE
2345							:CHECK BIT WORD
2346	013236	012700	000001	MOV	#1,R0		:SET FIRST BIT IN R1
2347	013242	012701	013053	MOV	#CHKCNT,R1		:ADDRESS OF C1 PARITY COUNTER
2348	013246	066701	167446	ADD	BASE,R1		
2349	013252	130067	177604	80\$: BITB	R0,PARBYT		:CHECKING FOR ODD OR EVEN PARITY
2350	013256	001401		BEQ	90\$:BRANCH IF EVEN PARITY
2351	013260	105211		INCB	(R1)		:ODD, INC THE PARITY COUNTER
2352	013262	132711	000001	90\$: BITB	#1,(R1)		:TEST PARITY COUNTER FOR SET BIT
2353	013266	001401		BEQ	100\$		
2354	013270	050004		BIS	R0,R4		:SET THE CHECK BIT IN CHECK
2355	013272	005201		100\$: INC	R1		:NEXT COUNTER
2356	013274	006300		ASL	R0		:SHIFT TEST BIT TO NEXT BIT
2357	013276	032700	000200	BIT	#200,R0		:CT YET?
2358	013302	001763		BEQ	80\$:BRANCH IF NO
2359	013304	000304		SWAB	R4		
2360	013306	006004		ROR	R4		
2361	013310	006004		ROR	R4		
2362	013312	006004		ROR	R4		:POSITION CHECK BITS IN BITS 11-5
2363	013314	010467	177474	MOV	R4,CHECK		
2364	013320	004767	000014	JSR	PC,RSTOT4		:RESTORE R0 TO R4
2365	013324	000207		RTS	PC		
2366							
2367							
2368				:CALL	JSR R4,SAVOT4		
2369	013326	010346		SAVOT4: MOV	R3,-(SP)		:SAVE R3
2370	013330	010246		MOV	R2,-(SP)		:SAVE R2
2371	013332	010146		MOV	R1,-(SP)		:SAVE R1
2372	013334	010046		MOV	R0,-(SP)		:SAVE R0


```

2373 013336 010407          MOV      R4,PC          ;R4 IS ALREADY SAVED
2374
2375          ;CALL      JSR      PC,RSTOT4
2376 013340 012604 RSTOT4: MOV      (SP)+,R4      ;RETURN ADDRESS
2377 013342 012600      MOV      (SP)+,R0      ;RESTORE R0
2378 013344 012601      MOV      (SP)+,R1      ;R1
2379 013346 012602      MOV      (SP)+,R2      ;R2
2380 013350 012603      MOV      (SP)+,R3      ;R3
2381 013352 000204      RTS       R4           ;RESTORE R4 AND RETURN
2382
2383
2384
2385
2386
2387 013354 010046          DASHL:  MOV      R0,-(SP)      ;SAVE R0
2388 013356 012500          MOV      (R5)+,R0          ;PICK UP ARGUMENT
2389 013360 066700 167334      ADD      BASE,R0          ;
2390 013364 006360 000002      ASL      2(R0)            ;SHIFT HIGH 16 BITS
2391 013370 006310          ASL      (R0)             ;SHIFT LOW 16 BITS
2392 013372 103002          BCC      10$              ;BR IF LOW WORD BIT 15 = 0
2393 013374 005260 000002      INC      2(R0)            ;IT WAS = 1 INC HI WORD
2394 013400 012600 10$:      MOV      (SP)+,R0          ;RESTORE R0
2395 013402 000205          RTS       R5
2396
2397 013404 012567 000112      BITCOM: MOV      (R5)+,MSKADR  ;FIRST ARG IS ADDR OF MASK
2398 013410 012567 000110      MOV      (R5)+,DATADR     ;SECOND IS ADDR OF DATA
2399 013414 010046          MOV      R0,-(SP)          ;
2400 013416 010146          MOV      R1,-(SP)          ;
2401 013420 010246          MOV      R2,-(SP)          ;
2402 013422 066767 167272 000072      ADD      BASE,MSKADR      ;SAVE THESE REGS
2403 013430 066767 167264 000066      ADD      BASE,DATADR      ;
2404 013436 017700 000060      MOV      @MSKADR,R0        ;GET MASK
2405 013442 017701 000056      MOV      @DATADR,R1        ;GET DATA
2406 013446 004767 000054      JSR      PC,XOR            ;DO THE XOR
2407 013452 010277 000046      MOV      R2,@DATADR        ;PUT RESULT BACK AS DATA
2408 013456 062767 000002 000036      ADD      #2,MSKADR         ;NEXT WORD
2409 013464 062767 000002 000032      ADD      #2,DATADR         ;DITTO
2410 013472 017700 000024      MOV      @MSKADR,R0        ;
2411 013476 017701 000022      MOV      @DATADR,R1        ;
2412 013502 004767 000020      JSR      PC,XOR            ;DO THE SAME FOR SECOND WORD
2413 013506 010277 000012      MOV      R2,@DATADR        ;PUT RESULT BACK TOO
2414 013512 012602          MOV      (SP)+,R2          ;
2415 013514 012601          MOV      (SP)+,R1          ;
2416 013516 012600          MOV      (SP)+,R0          ;RESTORE THE REGS R0,R2
2417 013520 000205          RTS       R5
2418
2419
2420 013522 000000      MSKADR: .WORD 0
2421 013524 000000      DATADR: .WORD 0
2422
2423
2424
2425          ;XOR ROUTINE
2426          ;* R0 = A OPER
2427          ;* R1 = B OPER
2428          ;* R2 = RESULT

```

2429
2430 013526 010102
2431 013530 040002
2432 013532 040100
2433 013534 050002
2434 013536 000207
2435
2436

XOR: MOV R1,R2 ;SAVE B OPERAND
BIC R0,R2 ;A NOT B
BIC R1,R0 ;B NOT A
BIS R0,R2 ;A XOR B
RTS PC ;EXIT AND RETURN

::*****


```

2437          ;* ERROR HANDLING ROUTINE
2438          ;* -----
2439          ;*
2440          ;*
2441          ;* PROGRAM COMES HERE EACH TIME AN ERROR IS ENCOUNTERED THIS
2442          ;* ROUTINE TYPES OUT THE ERROR MESSAGE IN THE FORMAT GIVEN EARLIER
2443
2444 013540 017637 000000 000402 ERROR: MOV @ (SP), @#$FATAL ;LOAD THE LOCATION $FATAL WITH THE ERROR NUMBER
2445 013546 005767 167152          TST ERRFLG ;FIRST ERROR ?
2446 013552 001050          BNE 1$ ;BR IF NOT FIRST ERROR
2447 013554 032777 020000 167150 BIT #20000, @SWR ;ERROR PRINTOUTS DESIRED ?
2448 013562 001044          BNE 1$ ;BR IN NO
2449 013564 004767 000674          JSR PC, TPCRLF ;TYPE ERROR HEADER FOR FIRST ERROR
2450 013570 040440 042104 020122 .ASCII / ADDR GOOD BAD PC ERR # PASFLG XOR /
2451 013576 020040 020040 047507
2452 013604 042117 020040 020040
2453 013612 041040 042101 020040
2454 013620 020040 020040 050040
2455 013626 020103 020040 020040
2456 013634 051105 020122 020043
2457 013642 020040 050040 051501
2458 013650 046106 020107 020040
2459 013656 054040 051117 020040
2460 013664 005015 000          .ASCIZ <15><12>
2461          013670          .EVEN
2462 013670 005267 167030          INC ERRFLG ;INDICATE NOT FIRST ERROR
2463 013674 010346          1$: MOV R3, -(SP) ;SAVE R3
2464 013676 010046          MOV R0, -(SP) ;AND R0 ON THE STACK
2465
2466          ;SETUP BANK NO. IN FATAL FOR APT
2467
2468 013700 010103          MOV R1, R3 ;GET VIRTUAL ADDRESS UNDER TEST FOR GETBNK
2469 013702 004767 001422          JSR PC, GETBNK ;GET BANK NO. UNDER TEST INTO PBANK
2470 013706 013703 000306          MOV @#PBANK, R3 ;GET BANK UNDER TEST
2471 013712 110337 000403          MOV B R3, @#$FATAL+1 ;STORE FAILING BANK NO. FOR APT
2472
2473          ;
2474
2475 013716 010346          MOV R3, -(SP) ;TEMPORARILY STORE R3
2476 013720 012703 000376          MOV #376, R3 ;MAKE R3 AS THE STACK POINTER
2477 013724 013743 000302          MOV @#PASFLG, -(R3) ;OUTPUT THE WORD STORED AT
2478 013730 005043          2$: CLR -(R3)
2479 013732 113713 000402          MOV B @#$FATAL, (R3) ;PUT ERROR NO. ON ERROR STACK
2480 013736 016643 000006          MOV 6(SP), -(R3) ;PLACE THE RETURN PC AT (R3)
2481 013742 011143          MOV (R1), -(R3) ;PLACE BAD DATA,
2482 013744 010043          MOV R0, -(R3) ;AND GOOD DATA ON THE STACK
2483 013746 005043          CLR -(R3)
2484 013750 016313 000004          MOV 4(R3), (R3) ;TAKE THE
2485 013754 040013          BIC R0, (R3) ;EXCLUSIVE OR OF GOOD AND BAD DATA
2486 013756 046300 000004          BIC 4(R3), R0 ;TO FIND THE BITS THAT FAILED
2487 013762 050013          BIS R0, (R3) ;AND PLACE IT ON THE STACK
2488 013764 011367 000746          MOV (R3), BDCHIP ;TO PRINT OUT XOR OF GOOD VS BAD
2489 013770 012700 002134          MOV #ENDPROG--24., R0 ;THIS CODE BRINGS THE RELATIVE ADDRESS
2490 013774 060700          ADD PC, R0 ;OF THE STARTING OF THE ERROR STACK
2491 013776 062700 000022          6$: ADD #18., R0 ;FOR THE SPECIFIC 4K BANK
2492 014002 005316          DEC (SP)

```



```

2549 014222 004767 001650          JSR    PC,CHECKC      ;IF CONTROL-C TYPED THEN PRINT ERROR HISTORY
2550                                ;AND HALT AT FATHLT.
2551 014226 105737 000042          7$:   TSTB   @#42      ;ARE WE RUNNING UNDER ACT?
2552 014232 001401                    BEQ    6$             ;BRANCH IF NO
2553
2554 014234 000777                    4$:   BR     4$             ;PROGRAM HAS HALTED ON ERROR, R1 IS POINTING
2555                                ;TO A LOCATION WHICH SHOULD HAVE CONTAINED
2556                                ;THE WORD STORED IN R0
2557 014236 012602                    6$:   MOV    (SP)+,R2    ;RESTORE R2
2558 014240 062716 000002          ADD    #2,(SP)        ;RESTORE THE RETURN ADDRESS
2559 014244 000207                    RTS    PC              ;RETURN FROM THE SUBROUTINE
2560
2561
2562
2563 014246                                FATERR:
2564 014246 004767 000212          SEQERR: JSR    PC,TPCRLF ;TYPE 'ERROR #'
2565 014252 051105 047522 020122    .ASCIZ /ERROR #/
2566 014260 000043                    .EVEN
2567
2568
2569 014262 017637 000000 000402    MOV    @ (SP),@#$FATAL ;LOAD THE LOCATION $FATAL WITH THE ERROR NUMBER
2570 014270 105237 000310          INCB   @#TYPCNT      ;TELL $TPNUM TO TYPE 1 WORD
2571 014274 012703 000376          MOV    #376,R3       ;$TPNUM USES R3 AS STACK
2572 014300 013743 000402          MOV    @#$FATAL,-(R3) ;PUT ERROR NO. ON STACK
2573 014304 005743                    TST    -(R3)         ;$TPNUM REQUIRES THIS
2574 014306 004767 000346          JSR    PC,FATYP      ;TYPE ERROR NO.
2575 014312 105737 000420          APTHLT: TSTB  @#$ENV   ;RUNNING UNDER APT?
2576 014316 001326                    BNE    FNDERR        ;BRANCH IF YES
2577 014320 000240          FATHLT: NOP
2578 014322 000776                    BR     FATHLT        ;FATAL ERROR OR ^C HALT.
2579                                ;RESTART AT 250 IF DESIRED
2580 014324 105737 000273          TPADER: TSTB  @#TYPENB
2581 014330 001406                    BEQ    1$             ;BR IF NO TYPE ENABLED
2582 014332 004767 000134          JSR    PC,PNTMES     ;PRINT MESSAGE ROUTINE
2583 014336 042101 020122 051105    .ASCIZ /ADR ERR/
2584 014344 000122                    .EVEN
2585
2586 014346 000207                    1$:   RTS    PC
2587
2588
2589
2590                                ;* TYPE OUT ROUTINE
2591                                ;* -----
2592                                ;*
2593                                ;* THIS ROUTINE IS USED BY THE PROGRAM TO TYPE OUT ANY CHARACTER
2594                                ;*
2595
2596 014350 010146 000002          NOTYP: MOV    R1,-(SP)
2597 014352 016601                    MOV    2(SP),R1
2598 014356 105721                    4$:   TSTB   (R1)+      ;IF THIS TYPE OUT HAS BEEN SUPRESSED THEN
2599 014360 001376                    BNE    4$             ;PREPARE TO RETURN
2600 014362 000412                    BR     RETTYP
2601 014364 010146          $TYPE: MOV    R1,-(SP)   ;SAVE R1
2602 014366 010046                    MOV    R0,-(SP)      ;AND R0 ON THE STACK
2603 014370 016601 000004          MOV    4(SP),R1      ;PLACE THE ADDRESS OF MESSAGE TO BE TYPED IN R1
2604 014374 112100                    2$:   MOVB   (R1)+,R0   ;PLACE THE BYTE TO BE TYPED IN R0

```

```
2605 014376 001403          BEQ      4$          ;IF IT IS END OF MESSAGE THEN GO TO 4$
2606 014400 004767 000022    JSR      PC,$STPCHR ;OTHERWISE GO TO TYPE THE CONTENTS OF RO
2607 014404 000773          BR       2$
2608 014406 012600          4$:      MOV      (SP)+,R0 ;RESTORE R0
2609 014410 005201    RETTYP: INC      R1      ;CAUSE R1 TO
2610 014412 042701 000001    BIC      #1,R1      ;POINT TO EVEN ADDRESS
2611 014416 010166 000002    MOV      R1,2(SP)   ;MODIFY THE RETURN ADDRESS
2612 014422 012601          MOV      (SP)+,R1  ;RESTORE R1
2613 014424 000416          BR       EXTYP      ;AND RETURN VIA RTS PC
2614
2615 014426 132737 000040 000421 $TPCHR: BITB   #40,@#$ENVM ;HAVE TYPE OUTS BEEN DISABLED?
2616 014434 001005          BNE      4$          ;IF SO THEN RETURN FROM THE SUBROUTINE
2617 014436 105737 177564    2$:      TSTB   @#$STPS ;WAIT HERE
2618 014442 100375          BPL      2$          ;UNTIL THE PRINTER IS READY
2619 014444 110037 177566    MOVB    RO,@#$TPB  ;LOAD DATA TO BE TYPED INTO DATA REG.
2620 014450 000404          BR       EXTYP      ;RETURN
2621
2622 014452 004767 177706    PCRLF:  JSR      PC,$TYPE ;CR/LF
2623 014456 005015          .ASCIZ  <15><12>
2624          .EVEN
2625 014462 000207    EXTYP:  RTS       PC      ;RETURN
2626
2627 014464 004767 177762    TPCRLF: JSR      PC,PCRLF ;TYPE CR/LF
2628 014470 000735          BR       $TYPE      ;NOW GO TO TYPE THE REST OF THE MESSAGE
2629
2630
2631 014472          CHGC2:
2632 014472 123737 000042 000046 PNTMES: CMPB   @#42,@#46 ;RUNNING UNDER ACT 11?
2633 014500 001723          BEQ     NOTYP      ;BRANCH IF YES -NOT PRINTOUT-
2634 014502 000770          BR     TPCRLF      ;SEND CR/LF AND TYPE MESSAGE.
```



```

2635
2636
2637
2638
2639
2640
2641
2642
2643 014504 004767 177742      TYPDEC: JSR    PC,PCRLF      ;TYPE CR/LF
2644
2645 014510 005046      $TPDEC: CLR    -(SP)
2646 014512 013746 000306      MOV    @#DECWRD,-(SP)      ;GET THE WORD THAT HAS TO BE CONVERTED TO A
2647
2648 014516 162716 000012      2$:    SUB    #10.,(SP)      ;DECIMAL NUMBER
2649 014522 002403
2650
2651 014524 005266 000002      BR
2652 014530 000772      INC    2(SP)                ;IF THE NUMBER IN (SP) WAS LESS THAN 10. THEN
2653 014532 062716 000012      4$:    BR    2$              ;GO TO 4$
2654 014536 052716 000060      ADD    #10.,(SP)            ;OTHERWISE ADD 1 TO THE LOCATION STORING 10'S DIGIT
2655 014542 112667 000020      BIS    #60,(SP)             ;AND RETURN TO 2$
2656 014546 052716 000060      MOVB   (SP)+,6$-2           ;MAKE THE CONTENTS OF (SP) A DECIMAL NUMBER
2657 014552 112667 000007      BIS    #60,(SP)             ;PLACE THE 1'S DIGIT TO BE TYPED
2658 014556 004767 177602      MOVB   (SP)+,6$-3           ;MAKE THE CONTENTS OF (SP) A DECIMAL NUMBER
2659
2660 014562 020040 030040 000060  .ASCIZ / 00/                ;PLACE THE 10'S DIGIT TO BE TYPED
2661
2662 014570 000207      6$:    JSR    PC,$TYPE        ;GO TO TYPE THE NUMBER IN DECIMAL FOLLOWED BY
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678 014572 032777 020000 166132  TYPERR: BIT    #20000,@SWR    ;ERROR PRINTOUT WANTED?
2679 014600 001055      BNE    OCTXT                ;BRANCH IF NO
2680 014602 004767 177644      JSR    PC,PCRLF            ;TYPE CR/LF
2681 014606 004767 000012      JSR    PC,TYPOCT           ;TYPE OCTAL NO.
2682 014612 000450      BR     OCTXT                ;RETURN VIA RTS PC
2683 014614 012123      OCTTYP: MOV    (R1)+,(R3)+   ;PLACE THE HIGH ORDER BITS AT LOCATION POINTED
2684
2685 014616 012113      MOV    (R1)+,(R3)          ;BY R3
2686 014620 105237 000310      INCB   @#TYPCNT            ;AND NOW PLACE THE LOW ORDER BITS
2687 014624 052743 000004      TYPOCT: BIS    #4,-(R3)     ;ENABLE THE TYPE OUT OF ONE OCTAL WORD
2688 014630 106113      2$:    ROLB   (R3)
2689 014632 103376      BCC    2$
2690 014634 005000      CLR    R0
  
```

```

;* OCTAL TYPE OUT ROUTINE
;* -----
;*
;* THIS ROUTINE IS USED TO TYPE OUT THE OCTAL VALUES
;* CONTROL SHOULD COME TO THIS ROUTINE WITH R3 POINTING TO
;* THE LOW ORDER BITS (I.E. BITS 0-15) OF THE ADDRESS TO
;* BE TYPED WHERE AS R3-2 SHOULD CONTAIN THE HIGH ORDER BITS
;* (I.E. BITS 16 & 17). CONTENTS OF LOCATION R3-1 AND R0 ARE
;* DESTROYED BY THIS SUBROUTINE
;* BYTE TYPCNT SHOULD BE SET TO THE NUMBER OF WORDS THAT HAVE
;* TO BE TYPED.
;*
  
```

2691	014636	106113		ROLB	(R3)		;GET BITS 17 & 16 INTO R0
2692	014640	006100		ROL	R0		
2693	014642	106113		ROLB	(R3)		
2694	014644	006100		ROL	R0		
2695	014646	000405		BR	\$TPNUM		
2696	014650	004767	177510	RPTOCT: JSR	PC,\$TYPE		; TYPE 3 SPACES
2697	014654	020040	000040	.ASCIZ	/ /		
2698				.EVEN			
2699	014660	005000		FATYP: CLR	R0		
2700	014662	012723	000006	\$TPNUM: MOV	#6,(R3)+		;ENABLE THE TYPE OUT OF 6 OCTAL DIGITS
2701	014666	000241		4\$: CLC			
2702	014670	006113		ROL	(R3)		
2703	014672	006100		ROL	R0		;PLACE THE CARRY FROM (R3) IN R0
2704	014674	052700	000060	BIS	#60,R0		;OR THE CONTENTS OF R0 WITH AN ASCII 0
2705	014700	004767	177522	JSR	PC,\$TPCHR		; TYPE THE OCTAL NUMBER STORED IN R0
2706	014704	005000		CLR	R0		
2707	014706	006113		ROL	(R3)		
2708	014710	006100		ROL	R0		;PLACE THE CARRY FROM (R3) IN R0
2709	014712	006113		ROL	(R3)		
2710	014714	006100		ROL	R0		;PLACE THE CARRY FROM (R3) IN R0
2711	014716	105363	177776	DECB	-2(R3)		;IF WE HAVEN'T TYPED THE 6 OCTAL DIGITS
2712	014722	001361		BNE	4\$;THEN REPEAT FROM 4\$
2713	014724	105337	000310	DECB	@#TYPCNT		;IF ALL THE WORDS REQUIRED HAVE NOT BEEN
2714	014730	100401		BMI	OCTXT		;BR IF = -1
2715	014732	001346		BNE	RPTOCT		;TYPED THEN REPEAT FROM RPTOCT
2716	014734			CHGC3:			
2717	014734	000207		OCTXT: RTS	PC		
2718							
2719	014736	000000		BDCHIP: .WORD	0		;XOR OF GOOD VS BAD DATA


```

2720
2721
2722
2723
2724
2725
2726
2727
2728
2729 014740 012702 001400      MEMMNG: MOV    #1400,R2
2730 014744 105037 000272      MMREG:  CLRB   @MMAVA          ;CLEAR THE BYTE THAT IS SUPPOSED TO INDICATE
2731                                     ;THAT MEM. MANAG. IS AVAILABLE FOR TESTING
2732 014750 032777 010000 165754  BIT    #10000,@SWR          ;HAS THE OPERATOR ASKED TO CHECK MEMORY MANAG. ?
2733 014756 001043                BNE    RETMM                ;IF NOT THEN RETURN FROM THE SUBROUTINE
2734 014760 012700 000004                MOV    #4,R0                ;PREPARE TO SETUP TIME OUT VECTOR
2735 014764 012720 015070                MOV    #NOMM,(R0)+          ;RETURN ADDRESS TO NOMM
2736 014770 012710 000340                MOV    #340,(R0)           ;AND WITH A PSW OF 340
2737 014774 005037 177572                CLR    @#SRO                ;TRY TO REACH MEM. MANAG. SRO
2738 015000 105237 000272                INCB   @MMAVA              ;IF IT IS AVAILABLE THEN SET MEM. MANAG. AVAILABLE
2739                                     ;BYTE
2740 015004 012701 172340                MOV    #172340,R1          ;R1 IS POINTING TO PAR0
2741 015010 005021                CLR    (R1)+                ;PAR0 WILL POINT TO BANK 0
2742 015012 012721 000200                MOV    #200,(R1)+          ;SET UP PAR 1
2743 015016 062702 000200      2$:   ADD    #200,R2
2744 015022 010221                MOV    R2,(R1)+            ;SETUP PAR1-PAR6
2745 015024 020127 172356                CMP    R1,#172356          ;ADDRESS OF PAR7
2746 015030 103772      2$:   BLO
2747 015032 012711 007600                MOV    #7600,(R1)          ;PAR7 IS POINTING TO THE I/O PAGE
2748 015036 012701 172300                MOV    #172300,R1
2749 015042 012721 077406      4$:   MOV    #77406,(R1)+        ;SETUP PDR0-PDR7
2750 015046 020127 172316                CMP    R1,#172316
2751 015052 101773      4$:   BLOS  4$
2752 015054 005237 177572                INC    @#SRO                ;ENABLE MEM. MANAG.
2753 015060 005010      $RETMM: CLR    (R0)                ;RESTORE TIME OUT TRAP VECTOR FOR ANY FUTURE TRAP
2754 015062 012740 000352                MOV    #BUSER,-(R0)
2755 015066 000207      RETMM: RTS    PC
2756
2757 015070 022626                NOMM:  CMP    (SP)+,(SP)+    ;RESTORE STACK POINTER
2758 015072 004767 177366                JSR    PC,TPCRLF           ;TYPE 'NO MEMORY MANAGEMENT MESSAGE
2759 015076 047516 046440 043516      .ASCIZ /NO MNG/
2760 015104      000
2761      015106
2762 015106 004767 177134      .EVEN
2763 015112 000104                JSR    PC,FATERR           ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
2764                                     ;*****ERROR NUMBER 104*****
2765 015114 000761                BR     $RETMM              ; RESTORE TIME OUT TRAP VECTOR
2766
2767 015116 013702 172354      UPMM:  MOV    @#172354,R2    ;PREPARE TO UPDATE MEMORY MANAG. REGISTERS
2768 015122 000710                BR     MMREG
  
```

2769
 2770
 2771
 2772
 2773
 2774
 2775
 2776
 2777
 2778
 2779

;* 18 BIT ADDRESS GENERATOR

 ;*

THIS SUBROUTINE IS USED TO PLACE THE ADDRESS STORED IN R1
 IN THE LOCATION POINTED BY R3. THE ADDRESS IN R1 IS CONVERTED
 TO AN 18 BIT ADDRESS ONLY IF MEM. MANAG. IS AVAILBLE IN WHICH
 CASE THE HIGH ORDER BITS OF THE ADDRESS ARE PLACED IN LOCATION
 POINTED BY R3-2
 ;*

2780 015124 005063 177776
 2781 015130 010113
 2782 015132 105737 000272
 2783 015136 001425
 2784 015140 010146
 2785 015142 042701 017777
 2786 015146 040113
 2787 015150 052701 004000
 2788 015154 006001
 2789 015156 103376
 2790 015160 062701 172340
 2791 015164 011101
 2792 015166 052701 010000
 2793 015172 006101
 2794 015174 103376
 2795 015176 006301
 2796 015200 006143
 2797 015202 006101
 2798 015204 006123
 2799 015206 050113
 2800 015210 012601
 2801 015212 000207

```

PUTADR: CLR      -2(R3)
          MOV      R1,(R3)          ;PLACE THE ADDRESS STORED IN R1 IN LOCATION (R3)
          TSTB    @MMMAVA          ;IS THE MEM. MANAG. AVAILBLE ?
          BEQ     6$                ;IF NOT THEN RETURN FROM THE SUBROUTINE
          MOV     R1,-(SP)          ;SAVE R1
          BIC     #17777,R1        ;CLEAR BITS 0-12 OF THE ADDRESS IN R1
          BIC     R1,(R3)          ;LEAVE BITS 0-12 OF THE ADDRESS IN (R3)
          BIS     #4000,R1         ;PREPARE TO SHIFT R1 BY 12 PLACES
2$:      ROR      R1
          BCC     2$                ;GET THE NUMBER OF PAR IN R1
          ADD     #172340,R1        ;GET THE ADDRESS OF PAR IN R1
          MOV     (R1),R1          ;LOAD R1 WITH THE CONTENTS OF PAR
          BIS     #10000,R1
4$:      ROL      R1
          BCC     4$                ;PLACE THE ADDRESS BITS 13-17 IN BITS 11-15 OF R1
          ASL     R1                ;SO WE DON'T PICK UP C BIT
          ROL     -(R3)            ;PLACE BIT 17 IN LOCATION POINTED BY R3-2
          ROL     R1
          ROL     (R3)+            ;PLACE BIT 16 OF THE ADDRESS
          BIS     R1,(R3)          ;PLACE BITS 13-15 OF THE ADDRESS IN LOCATION (R3)
          MOV     (SP)+,R1         ;RESTORE R1
6$:      RTS      PC                ;RETURN FROM THE SUBROUTINE
  
```

;* GET ADDRESS FROM THE APT MAILBOX

 ;*

THIS SUBROUTINE IS USED TO GET ADDRESS FROM APT MAILBOX AND
 PLACE IT IN THE LOCATION USED BY THE PROGRAM TO DEFINE THE
 MEMORY BOUNDRIES.
 PROGRAM CONTROL SHOULD COME TO THIS SUBROUTINE WITH R1 POINT-
 ING TO THE MEMORY TYPE IN THE APT MAILBOX AND R3 POINTING TO
 THE LOCATION+2 WHERE THE LOW ORDER BITS OF THE ADDRESS HAVE
 TO BE PLACED
 ;*

2814
 2815 015214 016143 000001
 2816 015220 005043
 2817
 2818 015222 116113 177777
 2819 015226 000207

```

GETADR: MOV      1(R1),-(R3)        ;PLACE THE LOW ORDER BITS OF THE ADDRESS
          CLR     -(R3)            ;CLEAR THE LOCATION WHERE THE HIGH ORDER BITS
          ;HAVE TO BE PLACED
2$:      MOVB    -1(R1),(R3)        ;PLACE BITS 16 & 17
          RTS      PC                ;RETURN FROM THE SUBROUTINE
  
```



```

2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830 015230 105237 000311    $GTSIZ: INCB    @#SAVKBB    ;PREPARE TO PLACE ADDRESS BITS 13-17 IN BITS
2831                                     ;0-4 OF R2
2832
2833 015234 012301    GETSIZ: MOV    (R3)+,R1
2834 015236 011302    MOV    (R3),R2    ;LOAD R2 WITH THE LOW ORDER BITS OF THE ADDRESS
2835 015240 042702 017777    BIC    #17777,R2    ;CLEAR ADDRESS BITS 0-12
2836 015244 052702 000040    2$:    BIS    #40,R2
2837 015250 006001    4$:    ROR    R1
2838 015252 006002    ROR    R2    ;ROTATE R1 AND R2 7 TIMES
2839 015254 103375    BCC    4$
2840 015256 105737 000311    TSTB   @#SAVKBB
2841 015262 001405    BEQ    6$
2842 015264 105037 000311    CLRB   @#SAVKBB
2843 015270 052702 000100    BIS    #100,R2
2844 015274 000765    BR     4$
2845 015276 012301    6$:    MOV    (R3)+,R1    ;PLACE THE LOW ORDER ADDRESS BITS IN R1
2846 015300 012700 160000    MOV    #160000,R0
2847 015304 040001    BIC    R0,R1    ;LEAVE BITS 0-12 OF THE ADDRESS IN R1
2848 015306 000207    RTS    PC    ;RETURN FROM THE SUBRORNE
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858 015310 105737 000272    CLRMM: TSTB   @#MMAVA    ;WAS THE MEMORY MANAGEMENT ENABLED ?
2859 015314 001404    BEQ    1$    ;IF NOT THEN GO TO 1$
2860 015316 005037 177572    CLR    @#SRO    ;DISABLE THE MEMORY MANAGEMENT
2861 015322 105037 000272    CLRB   @#MMAVA    ;AND DO NOT ATTEMPT TO TEST MEM. MANAG.
2862 015326 000207    1$:    RTS    PC    ;RETURN FROM THE SUBROUTINE
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872 015330 010046    ;* GET BANK NO. UNDER TEST
2873 015332 010346    ;CALLED BY ERRYP TO GET BANK NO. UNDER TEST INTO PBNK.
2874 015334 042703 017777    ;REGISTERS
2875 015340 052703 010000    ;R0=POINTER TO PAR UNDER TEST
    ;R3=VIRTUAL ADDRESS ON ENTRY
    ;R0+R3 ARE RESTORED ON EXIT.
    GETBNK: MOV    R0,-(SP)    ;SAVE R0
    MOV    R3,-(SP)    ;SAVE R3
    BIC    #17777,R3    ;SAVE ONLY VIRTUAL BANK BITS
    BIS    #10000,R3    ;SETUP R3 SHIFT BIT

```

```

2876 015344 000241
2877 015346 006003
2878 015350 103376
2879 015352 105737 000272
2880 015356 001407
2881
2882
2883 015360 006303
2884 015362 062703 172340
2885 015366 011300
2886 015370 006300
2887 015372 000300
2888 015374 110003
2889 015376 010337 000306
2890 015402 012603
2891 015404 012600
2892 015406 000207
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902 015410 010146
2903 015412 042701 003777
2904 015416 052701 002000
2905 015422 000241
2906 015424 006001
2907 015426 103376
2908 015430 010100
2909 015432 105737 000272
2910 015436 001416
2911
2912 015440 010146
2913 015442 006201
2914 015444 042701 000001
2915 015450 062701 172340
2916 015454 011100
2917 015456 006300
2918 015460 006300
2919 015462 006300
2920 015464 000300
2921 015466 042716 000034
2922 015472 052600
2923 015474 012601
2924 015476 000207
2925
2926
2927
2928
2929
2930
2931

1$: CLC
ROR R3 ;SHIFT A BANK BIT
BCC 1$ ;UNTIL IN BITS <2:0> OF R3
TSTB @#MMAVA ;MEMORY MANAGEMENT UNDER TEST?
BEQ 2$ ;NO EXIT

;GET PAR ADDRESS AND PHYSICAL BANK NO.
ASL R3 ;MAKE R3 PAR ADDRESS OFFSET.
ADD #172340,R3 ;MAKE FULL PAR ADDRESS.
MOV (R3),R0 ;GET PAR CONTENTS
ASL R0
SWAB R0 ;SHIFT BANK BITS TO BITS <7:0>
MOVB R0,R3 ;SET R3 TO PHYSICAL BANK NO.
2$: MOV R3,@#PBANK ;STORE PHYSICAL BANK NO.
MOV (SP)+,R3 ;RESTORE R3
MOV (SP)+,R0 ;RESTORE R0
RTS PC ;RETURN TO CALLER

;GET THE 1K BANK UNDER TEST
;CALLED BY TEST 7
;REGISTERS
; R1 = VIRTUAL ADDRESS ON ENTRY
; R0 = POINTER TO 1K UNDER TEST ON EXIT
;R1 WILL BE RESTORED ON EXIT

GET1K: MOV R1,-(SP) ;SAVE R1
BIC #3777,R1 ;ONLY WANT 1K BIT COUNT
BIS #2000,R1 ;SET UP SHIFT BIT
1$: CLC ;CLEAR CARRY BIT
ROR R1 ;ROTATE THE BANK
BCC 1$ ;UNTIL THEYR'E IN BITS 4:0
MOV R1,R0 ;IN CASE NO MEM MGNT
TSTB @#MMAVA ;MEM MGNT AVAILABLE ?
BEQ 2$ ;BR IF NO

;GET PAR AND PHYSICAL BANK
MOV R1,-(SP) ;SAVE 1K BANK
ASR R1 ;MAKE R1 A PAR ADDRESS
BIC #1,R1 ;PAR ADDRESSES ARE 4K BOUNDARIES
ADD #172340,R1 ;MAKE IT A FULL PAR ADDR.
MOV (R1),R0 ;GET THE PAR CONTENTS
ASL R0 ;SHIFT R0
ASL R0
ASL R0
SWAB R0
BIC #34,(SP) ;ONLY WANT THE 2 LSB'S
BIS (SP)+,R0 ;SET THE IN PAR
2$: MOV (SP)+,R1 ;RESTORE R1
RTS PC ;AND EXIT

;THIS ROUTINE CHECKS THE ADDRESS IN R1
;TO MAKE SURE IT IS ERROR FREE, INCLUDING
;THE CHECKBITS.
;R3 = 0 IF NO ERROR
;R3 = NONZERO IF ERROR IS FOUND

```



```

2932
2933 015500 016777 165204 165222 TSTADD: MOV    ECCDIS,@CSRADR ;DISABLE ECC
2934 015506 005011          CLR    (R1)          ;CLEAR A LOC
2935 015510 005061 000002          CLR    2(R1)        ;UPPER WORD TOO
2936 015514 005711          TST    (R1)          ;READ ZEROS
2937 015516 001047          BNE    1$           ;BR IF ERROR
2938 015520 005761 000002          TST    2(R1)        ;SHOULD BE ZEROS ALSO
2939 015524 001044          BNE    1$           ;
2940 015526 005111          COM    (R1)          ;COMPLEMENT OF 0 = 177777
2941 015530 005161 000002          COM    2(R1)        ;BOTH WORDS
2942 015534 022711 177777          CMP    #-1,(R1)     ;READ ALL ONES
2943 015540 001036          BNE    1$           ;BR IF NO
2944 015542 022761 177777 000002          CMP    #-1,2(R1)   ;READ AGAIN
2945 015550 001032          BNE    1$           ;BR IF ERROR
2946 015552 016703 165152          MOV    CSRADR,R3    ;GET ADDRESS OF CSR
2947 015556 016713 165130          MOV    DIAGA,(R3)   ;SET DIAGNOSTIC BIT
2948 015562 052713 000002          BIS    #2,(R3)     ;AND ECCDIS BIT
2949 015566 012711 000000          MOV    #0,(R1)     ;TO WRITE ALL ZERO CHECKBITS
2950 015572 005761 000002          TST    2(R1)        ;READ AND RECORD CHECKBITS
2951 015576 032713 007740          BIT    #7740,(R3)  ;NO CHECKBIT SHOULD BE SET
2952 015602 001015          BNE    1$           ;BR IF SET
2953 015604 052713 007740          BIS    #7740,(R3)  ;MAKE CHECKBITS ALL ONES
2954 015610 012711 177777          MOV    #-1,(R1)    ;WRITE THEM LIKE THIS
2955 015614 042713 007740          BIC    #7740,(R3)  ;SO WE KNOW IF WE READ THEM BACK
2956 015620 005761 000002          TST    2(R1)        ;READ THEM
2957 015624 042713 170037          BIC    #170037,(R3);ONLY WANT CHECKBITS
2958 015630 022713 007740          CMP    #7740,(R3)  ;SHOULD BE ALL ONES
2959 015634 001403          BEQ    2$           ;BR IF OK
2960 015636 012703 000001          1$: MOV    #1,R3      ;INDICATE ERROR FOUND
2961 015642 000401          BR     3$           ;AND EXIT
2962 015644 005003          2$: CLR    R3        ;CLEARED = GOOD LOC
2963 015646 005077 165056          3$: CLR    @CSRADR   ;CLEAR CSR BEFORE LEAVING
2964 015652 000207          RTS    PC           ;AND EXIT
2965
2966
2967          ;THIS ROUTINE IS USED TO COUNT THE BAD CHECKBITS
2968          ;FROM TEST 15. IT USES THE ROUTINE 'XOR' AND THEN
2969          ;COUNTS THE BAD ONES
2970
2971 015654 005067 000056          BITCHK: CLR    BITCNT ;WANT TO START CLEAN
2972 015660 010146          MOV    R1,-(SP)    ;SAVE ADDRESS
2973 015662 017701 165042          MOV    @CSRADR,R1 ;GET CONTENTS OF CSR
2974 015666 042701 170037          BIC    #170037,R1 ;ONLY WANT CHECK BITS
2975 015672 010146          MOV    R1,-(SP)    ;SAVE THIS TOO
2976 015674 004767 175626          JSR    PC,XOR      ;RO XOR R1 = R2
2977 015700 012701 000020          MOV    #20,R1     ;GETTING READY TO COUNT
2978
2979
2980 015704 006301          1$: ASL    R1        ;SHIFT TO NEXT POSITION
2981 015706 020127 010000          CMP    R1,#10000  ;FINISHED YET ?
2982 015712 001405          BEQ    2$           ;BR IF DONE
2983 015714 030102          BIT    R1,R2       ;SEE IF THIS BIT (R1) IS BAD
2984 015716 001772          BEQ    1$           ;BR IF NOT BAD
2985 015720 005267 000012          INC    BITCNT     ;IT IS, COUNT IT
2986 015724 000767          BR     1$           ;NOT DONE YET, KEEP GOING
2987 015726 012600          2$: MOV    (SP)+,R0 ;FIRST GET DATA

```

```

2988 015730 012601          MOV      (SP)+,R1          ;THEN ADDRESS
2989 015732 010011          MOV      R0,(R1)          ;PUT BAD DATA AWAY FOR PRINTOUT
2990 015734 000207          RTS      PC                ;THEN EXIT
2991
2992 015736 000000          BITCNT: .WORD 0           ;# OF BAD CHECKBITS FOUND ABOVE
2993
2994
2995
2996
2997 015740 004767 176520          ;THIS ROUTINE PRINTS THE TITLE AND MESSAGE
2998 015744 055103 046515 041514  PRTITL: JSR      PC,TPCRLF          ;PRINT ROUTINE
2999 015752 020060 043115 046114          .ASCII  /CZMMLCO MFLLS-K MEM DIAG/
3000 015760 026523 020113 042515
3001 015766 020115 044504 043501
3002 015774 052440 042523 052040          .ASCII  / USE TO TEST MF11S-K ECC MEMORY/
3003 016002 020117 042524 052123
3004 016010 046440 030506 051461
3005 016016 045455 042440 041503
3006 016024 046440 046505 051117
3007 016032          131
3008 016033          015 052012 050131          .ASCIIZ <15><12>/TYPE <CONTROL C> TO EXIT/
3009 016040 020105 041474 047117
3010 016046 051124 046117 041440
3011 016054 020076 047524 042440
3012 016062 044530 000124
3013
3014 016066 012767 172136 164634          .EVEN
3015 016074 000207          MOV      #172136,CSRADR          ;ORIGINAL CSR ADDRESS
3016
3017
3018
3019
3020
3021
3022
3023
3024
3025
3026
3027
3028 016076 105037 000311          ;CHECKC
3029 016102 105737 177560          ; THIS ROUTINE CHECKS IF CONTROL-C WAS TYPED AT THE END OF EACH
3030 016106 100021          ;TEST OR IN THE ERROR TYPE ROUTINE.
3031 016110 113702 177562          ;IF CONTROL-C TYPED THE PROGRAM IS RETURNED TO LOWER MEMORY IF IT WAS
3032 016114 042702 000200          ;RELOCATED AND THE ERROR HISTORY IS PRINTED OUT.
3033 016120 122702 000003          ;FINALLY IT HALTS AT FATHLT.
3034 016124 001012
3035 016126 110237 000311          CHECKC: CLRB      @#SAVKBB          ;INIT CONTROL-C FLAG.
3036 016132 004767 176326          TSTB      @#TKS                ;ANY CHAR. TYPED?
3037 016136 041536          000          BPL      EXITC                ;BR IF NO-EXIT VIA RTS PC-
3038
3039 016142 005067 164556          MOV      @#SKBB,R2            ;GET THE CHAR TYPED.
3040 016146 000167 173730          BIC      #200,R2              ;CLEAR THE PARITY BIT.
3041 016152 000207          CMPB     #3,R2                ;IS IT CONTROL-C?
3042
3043 016154 000000          BNE      EXITC                ;BRANCH IF NO -EXIT VIA RTS PC-
3044
3045
3046
3047
3048
3049
3050
3051
3052
3053
3054
3055
3056
3057
3058
3059
3060
3061
3062
3063
3064
3065
3066
3067
3068
3069
3070
3071
3072
3073
3074
3075
3076
3077
3078
3079
3080
3081
3082
3083
3084
3085
3086
3087
3088
3089
3090
3091
3092
3093
3094
3095
3096
3097
3098
3099
3100
3101
3102
3103
3104
3105
3106
3107
3108
3109
3110
3111
3112
3113
3114
3115
3116
3117
3118
3119
3120
3121
3122
3123
3124
3125
3126
3127
3128
3129
3130
3131
3132
3133
3134
3135
3136
3137
3138
3139
3140
3141
3142
3143
3144
3145
3146
3147
3148
3149
3150
3151
3152
3153
3154
3155
3156
3157
3158
3159
3160
3161
3162
3163
3164
3165
3166
3167
3168
3169
3170
3171
3172
3173
3174
3175
3176
3177
3178
3179
3180
3181
3182
3183
3184
3185
3186
3187
3188
3189
3190
3191
3192
3193
3194
3195
3196
3197
3198
3199
3200
3201
3202
3203
3204
3205
3206
3207
3208
3209
3210
3211
3212
3213
3214
3215
3216
3217
3218
3219
3220
3221
3222
3223
3224
3225
3226
3227
3228
3229
3230
3231
3232
3233
3234
3235
3236
3237
3238
3239
3240
3241
3242
3243
3244
3245
3246
3247
3248
3249
3250
3251
3252
3253
3254
3255
3256
3257
3258
3259
3260
3261
3262
3263
3264
3265
3266
3267
3268
3269
3270
3271
3272
3273
3274
3275
3276
3277
3278
3279
3280
3281
3282
3283
3284
3285
3286
3287
3288
3289
3290
3291
3292
3293
3294
3295
3296
3297
3298
3299
3300
3301
3302
3303
3304
3305
3306
3307
3308
3309
3310
3311
3312
3313
3314
3315
3316
3317
3318
3319
3320
3321
3322
3323
3324
3325
3326
3327
3328
3329
3330
3331
3332
3333
3334
3335
3336
3337
3338
3339
3340
3341
3342
3343
3344
3345
3346
3347
3348
3349
3350
3351
3352
3353
3354
3355
3356
3357
3358
3359
3360
3361
3362
3363
3364
3365
3366
3367
3368
3369
3370
3371
3372
3373
3374
3375
3376
3377
3378
3379
3380
3381
3382
3383
3384
3385
3386
3387
3388
3389
3390
3391
3392
3393
3394
3395
3396
3397
3398
3399
3400
3401
3402
3403
3404
3405
3406
3407
3408
3409
3410
3411
3412
3413
3414
3415
3416
3417
3418
3419
3420
3421
3422
3423
3424
3425
3426
3427
3428
3429
3430
3431
3432
3433
3434
3435
3436
3437
3438
3439
3440
3441
3442
3443
3444
3445
3446
3447
3448
3449
3450
3451
3452
3453
3454
3455
3456
3457
3458
3459
3460
3461
3462
3463
3464
3465
3466
3467
3468
3469
3470
3471
3472
3473
3474
3475
3476
3477
3478
3479
3480
3481
3482
3483
3484
3485
3486
3487
3488
3489
3490
3491
3492
3493
3494
3495
3496
3497
3498
3499
3500
3501
3502
3503
3504
3505
3506
3507
3508
3509
3510
3511
3512
3513
3514
3515
3516
3517
3518
3519
3520
3521
3522
3523
3524
3525
3526
3527
3528
3529
3530
3531
3532
3533
3534
3535
3536
3537
3538
3539
3540
3541
3542
3543
3544
3545
3546
3547
3548
3549
3550
3551
3552
3553
3554
3555
3556
3557
3558
3559
3560
3561
3562
3563
3564
3565
3566
3567
3568
3569
3570
3571
3572
3573
3574
3575
3576
3577
3578
3579
3580
3581
3582
3583
3584
3585
3586
3587
3588
3589
3590
3591
3592
3593
3594
3595
3596
3597
3598
3599
3600
3601
3602
3603
3604
3605
3606
3607
3608
3609
3610
3611
3612
3613
3614
3615
3616
3617
3618
3619
3620
3621
3622
3623
3624
3625
3626
3627
3628
3629
3630
3631
3632
3633
3634
3635
3636
3637
3638
3639
3640
3641
3642
3643
3644
3645
3646
3647
3648
3649
3650
3651
3652
3653
3654
3655
3656
3657
3658
3659
3660
3661
3662
3663
3664
3665
3666
3667
3668
3669
3670
3671
3672
3673
3674
3675
3676
3677
3678
3679
3680
3681
3682
3683
3684
3685
3686
3687
3688
3689
3690
3691
3692
3693
3694
3695
3696
3697
3698
3699
3700
3701
3702
3703
3704
3705
3706
3707
3708
3709
3710
3711
3712
3713
3714
3715
3716
3717
3718
3719
3720
3721
3722
3723
3724
3725
3726
3727
3728
3729
3730
3731
3732
3733
3734
3735
3736
3737
3738
3739
3740
3741
3742
3743
3744
3745
3746
3747
3748
3749
3750
3751
3752
3753
3754
3755
3756
3757
3758
3759
3760
3761
3762
3763
3764
3765
3766
3767
3768
3769
3770
3771
3772
3773
3774
3775
3776
3777
3778
3779
3780
3781
3782
3783
3784
3785
3786
3787
3788
3789
3790
3791
3792
3793
3794
3795
3796
3797
3798
3799
3800
3801
3802
3803
3804
3805
3806
3807
3808
3809
3810
3811
3812
3813
3814
3815
3816
3817
3818
3819
3820
3821
3822
3823
3824
3825
3826
3827
3828
3829
3830
3831
3832
3833
3834
3835
3836
3837
3838
3839
3840
3841
3842
3843
3844
3845
3846
3847
3848
3849
3850
3851
3852
3853
3854
3855
3856
3857
3858
3859
3860
3861
3862
3863
3864
3865
3866
3867
3868
3869
3870
3871
3872
3873
3874
3875
3876
3877
3878
3879
3880
3881
3882
3883
3884
3885
3886
3887
3888
3889
3890
3891
3892
3893
3894
3895
3896
3897
3898
3899
3900
3901
3902
3903
3904
3905
3906
3907
3908
3909
3910
3911
3912
3913
3914
3915
3916
3917
3918
3919
3920
3921
3922
3923
3924
3925
3926
3927
3928
3929
3930
3931
3932
3933
3934
3935
3936
3937
3938
3939
3940
3941
3942
3943
3944
3945
3946
3947
3948
3949
3950
3951
3952
3953
3954
3955
3956
3957
3958
3959
3960
3961
3962
3963
3964
3965
3966
3967
3968
3969
3970
3971
3972
3973
3974
3975
3976
3977
3978
3979
3980
3981
3982
3983
3984
3985
3986
3987
3988
3989
3990
3991
3992
3993
3994
3995
3996
3997
3998
3999
4000

```


CZMMLCO MF11S-K MEM DIAG
CZMMLC.P11 17-MAR-80 10:47

MACY11 30A(1052) 17-MAR-80 10:48 ^{C 6} PAGE 60
SUBROUTINE TO DISABLE MEMORY MANAGEMENT

SEQ 0067

3044
3045
3046
3047

000001

.END

;STACK.FOR EACH 4K BANK 18. BYTES ARE SAVED.
;ALSO THE ABSOLUTE LOADER AND XXDP CODE IS SAVED
;AFTER THE ERROR STACK.

END13	010364	1710#												
END14	010476	1748#												
END15	011330	1880#												
END16	011500	1964#												
END17	011636	2015#												
END2	003540	903#												
END3	003772	966#												
END4	004146	1012#												
END5	004616	1100	1112	1115#										
END6	005534	1125	1254#											
END7	006526	1411#												
ERRFLG	002724	711#	2237*	2445	2462*	3039*								
ERROR	013540	765	795	802	823	830	871	878	887	894	935	955	1005	1057
		1064	1071	1079	1088	1096	1173	1196	1202	1210	1229	1307	1319	1364
		1463	1470	1551	1558	1637	1644	1696	1740	1760	1786	1808	1829	1853
		1871	1904	1911	1938	1990	2444#							
ERRTYP	014010	2497#												
EXITC	016152	3030	3034	3041#										
EXTYP	014462	2613	2620	2625#										
FAILNM	012614	2206	2210#											
FATERR	014246	321	357	402	557	575	598	987	2052	2563#	2762			
FATHLT	014320	2541	2577#	2578										
FATYP	014660	2574	2699#											
FNDERR	014174	2538#	2576											
GETADR	015214	406	407	2815#										
GETBNK	015330	2469	2872#											
GETSIZ	015234	2114	2126	2131	2833#									
GET1K	015410	1357	2902#											
HIGHAD	000330	212#	393											
HIGHTW	000326	211#	392	510										
INHREL	002726	316*	324	330*	335	712#	2033							
LDFLG	002716	295*	311*	437	708#	993	2029							
LOOP	002460	638#	696											
LOWADD	000324	209#	414	439*										
LOWBNK	000300	175#	176											
LOWER	012200	2094	2098#											
LOWTWO	000322	208#	440*	501	549	2113								
M =	000200	23#												
MAXADR	012346	2129	2132	2143#										
MAXMEM	000336	218#	391	522*										
MEMPING	014740	454	2105	2729#										
MEMTST	002224	582#												
MINMEM	000320	41*	42*	204#	445*	547	602	624*	787	916	922	1043	1150	1178
		1215	1235	1266	1451	1457	1537	1543	1547	1580	1624	1633	1668	1766
		2061*	2084*	2125*	2163*									
MMAVA	000272	157#	160	435	455	2026	2107	2730*	2738*	2782	2858	2861*	2879	2909
MMREG	014744	2124	2730#	2768										
MSKADR	013522	2397*	2402*	2404	2408*	2410	2420#							
MSYES	002714	413*	431	441*	444*	450	707#							
N =	000105	23#	321	324#	357	360#	402	405#	557	560#	575	578#	598	601#
		736	739#	750	752#	753	756#	761	763#	784	787#	795	798#	802
		805#	823	826#	830	833#	857	860#	871	874#	878	881#	887	890#
		894	897#	911	914#	935	938#	955	958#	976	979#	986	990#	1005
		1008#	1022	1025#	1057	1060#	1064	1067#	1071	1074#	1079	1082#	1088	1091#
		1096	1099#	1121	1124#	1173	1176#	1202	1205#	1210	1213#	1262	1265#	1307
		1310#	1319	1322#	1364	1367#	1420	1423#	1463	1466#	1470	1473#	1506	1509#

CZMMLCO MF11S-K MEM DIAG
CZMMLC.P11 17-MAR-80 10:47

MACY11 30A(1052) 17-MAR-80 10:48 L 6 PAGE 71
CROSS REFERENCE TABLE -- MACRO NAMES

SEQ 0076

.\$READ 1#
.\$R2AZ 1#
.\$SAVE 1#
.\$SB2D 1#
.\$SB2O 1#
.\$SCOP 1#
.\$SIZE 1#
.\$SUPR 1#
.\$STRAP 1#
.\$TYPB 1#
.\$TYPD 1#
.\$TYPE 1#
.\$TYPO 1#
.\$4OCA 1#
.\$1170 1#

. ABS. 016156 000

ERRORS DETECTED: 0

CZMMLC.BIN,CZMMLC.LST/CRF/SOL/NL:TOC=CZMMLC.SML,CZMMLC.P11
RUN-TIME: 31 42 2 SECONDS
RUN-TIME RATIO: 153/76=1.9
CORE USED: 32K (63 PAGES)