

1 .REPT 0

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

IDENTIFICATION

PRODUCT CODE: AC-8850F-MC
PRODUCT NAME: CZKMAF0 MOS/CORE 0-124K EXER
DATE CREATED: MAR., 1979
MAINTAINER: DIAGNOSTIC GROUP

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

THE SOFTWARE DESCRIBED IN THIS DOCUMENT IS FURNISHED TO THE PURCHASER UNDER A LICENSE FOR USE ON A SINGLE COMPUTER SYSTEM AND CAN BE COPIED (WITH INCLUSION OF DIGITALS COPYRIGHT NOTICE) ONLY FOR USE IN SUCH SYSTEM, EXCEPT AS MAY OTHERWISE BE PROVIDED IN WRITING BY DIGITAL.

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

COPYRIGHT (C) 1975,1979 DIGITAL EQUIPMENT CORPORATION

| | CONTENTS | |
|-----|----------|---|
| | ----- | |
| 49 | | |
| 50 | | |
| 51 | | |
| 52 | | |
| 53 | 1.0 | ABSTRACT |
| 54 | 1.1 | GETTING STARTED |
| 55 | | |
| 56 | 2.0 | REQUIREMENTS |
| 57 | 2.1 | EQUIPMENT |
| 58 | 2.2 | STORAGE |
| 59 | | |
| 60 | 3.0 | LOADING PROCEDURE |
| 61 | | |
| 62 | 4.0 | STARTING PROCEDURE |
| 63 | 4.1 | SWITCH SETTINGS |
| 64 | 4.2 | CONTROL-C OPTION |
| 65 | 4.3 | STARTING ADDRESS =200 |
| 66 | | RESTART ADDRESS =250 |
| 67 | 4.4 | PROGRAM AND/OP OPERATOR ACTION |
| 68 | 4.5 | LONG GALLOP OPTION |
| 69 | | |
| 70 | 5.0 | PROGRAM HALTS (NORMAL + ERROR) |
| 71 | | |
| 72 | 6.0 | ERRORS |
| 73 | 6.1 | ERROR MESSAGE FORMAT. |
| 74 | 6.2 | ERROR DICTIONARY |
| 75 | 6.3 | ERROR HISTORY |
| 76 | 6.4 | ERROR RECOVERY |
| 77 | | |
| 78 | 7.0 | RESTRICTIONS |
| 79 | | |
| 80 | 8.0 | MISCELLANEOUS |
| 81 | 8.1 | ADDRESS/BANK RANGES IN OCTAL AND DECIMAL |
| 82 | 8.2 | EXECUTION TIME |
| 83 | 8.3 | PASS COUNT AND TEST NO. LOCATIONS |
| 84 | 8.4 | STACK POINTER |
| 85 | 8.6 | POWER FAIL |
| 86 | | |
| 87 | 9.0 | PROGRAM DESCRIPTION |
| 88 | 9.1 | NARRATIVE FLOW CHART |
| 89 | 9.2 | TEST TITLES |
| 90 | | TEST 0: TEST FOR PROPER BANK SELECTION |
| 91 | | TEST 1: CHECK DATI/DATO LINES |
| 92 | | TEST 2: TEST MEMORY FOR HOLDING DATA AND BYTE SELECTION |
| 93 | | TEST 3: DUAL ADDRESS TEST A |
| 94 | | TEST 4: DUAL ADDRESS TEST B |
| 95 | | TEST 5: MARCHING 1'S AND 0'S |
| 96 | | TEST 6: CELLS' VOLATILITY TEST |
| 97 | | TEST 7: SHIFTING DIAGONAL |
| 98 | | TEST 10: READ RECOVERY GALLOPING TEST THROUGH EVERY 64TH CELL |
| 99 | | TEST 11: READ RECOVERY LONG GALLOPING/FAST GALLOPING TEST |
| 100 | | TEST 12: WORST CASE TESTING FOR CORE MEMORY |
| 101 | | TEST 13: WRITE RECOVERY TEST |
| 102 | | |
| 103 | 10.0 | RXDP & ACT11 & APT OPERATION |

104 [1.0] ABSTRACT

105
106 THIS DIAGNOSTIC WILL TEST 0 - 124K OF MOS OR CORE MEMORY
107 ON ANY PDP-11 FAMILY COMPUTER. SOME TESTS ARE WORST CASE
108 FOR MOS AND SOME FOR CORE, BUT ALL TESTS ARE ALWAYS RUN.
109 THE TESTS OCCUPIES LESS THAN 2K OF MEMORY SO IT CAN BE
110 USED TO TEST A SYSTEM WITH ONLY 4K OF MEMORY. IF ONLY 4K
111 EXISTS, HOWEVER, THE ABSOLUTE LOADER IS NOT SAVED.

112
113 THIS PROGRAM CAN BE RUN UNDER XXDP, APT AND ACT MONITORS.
114 ON PROCESSORS WITH NO HARDWARE SWITCH REGISTER, SOFTWARE
115 SWITCH REGISTER = LOCATION 176.
116

117 [1.1] GETTING STARTED

118
119 IF NO HARDWARE SWITCH REGISTER SET LOCATION 176 TO OBTAIN SWITCH
120 OPTIONS.

121 TO START:

122 -----

- 123
124
125 A. SET SWITCH REGISTER = 00000
126 B. START AT 200.
127 C. THE MEMORY LIMITS WILL BE PRINTED.
128 D. SEE SECTION 4.4 FOR REST OF PRINTOUTS EXPECTED.
129 E. "PASS#01" WILL BE TYPED LAST, AND THE TEST WILL
130 RESTART.
131 F. TO HALT THE TEST, TYPE CONTROL-C, THIS WILL INSURE THE
132 PROGRAM IS RELOCATED BACK TO LOWER MEMORY.
133 BE PATIENT, THE CONTROL-C IS ONLY RECOGNIZED AT THE END
134 OF THE CURRENT SUBTEST.
135 G. IF AN UNEXPECTED HALT OCCURS SEE SECTION 6.0. IF AN
136 ERROR # IS TYPED SEE SECTION 6.2.

137
138 [CAUTION! BEFORE "DIGGING" INTO THE LISTING READ
139 SECTION 9.

140
141 SWITCH SETTING SUMMARY (SEE SECTION 4.1 FOR DETAILS)

142 -----

143
144 BIT15(100000) HALT ON ERROR
145 BIT14(040000) LOOP IN SUBTEST DEFINED BY BITS <310>
146 BIT13(020000) INHIBIT ERROR PRINTOUTS
147 BIT12(010000) ENABLE TESTING ABOVE 28K (WITH MEMORY MANAGEMENT)
148 BIT11(004000) ENABLE PARITY TESTING
149 BIT10(002000) HALT AFTER EACH SUBTEST
150 BIT09(001000) INHIBIT PROGRAM RELOCATION
151 BIT08(000400) TYPE FIRST FAILING BIT ERROR PER 4K.
152 BIT07(000200) ENABLE LONG GALLOPING TEST
153 BIT06(000100) INHIBIT MEMORY SIZING
154 BIT05(000040) INHIBIT "PASS#XX" PRINTOUTS
155 BIT04(000020) INHIBIT PRINTOUTS
156 BIT03-BIT00 BEGINNING TEST NUMBER.
157
158
159

160 [2.0] REQUIREMENTS
161
162 [2.1] EQUIPMENT
163
164 STANDARD 11 FAMILY COMPUTER WITH A CONSOLE OUTPUT DEVICE
165 AND FROM 4K TO 124K OF MEMORY. PROGRAM WILL ALSO RUN ON THE
166 PDP-11 AND ON 30K LSI SYSTEMS.
167
168
169
170 [2.2] STORAGE
171
172 PROGRAM STORAGE = 0000 = 7744. PROGRAM EXPANDS FOR ERROR
173 HISTORY AND TO SAVE ABSOLUTE LOADER OR XXDP CHAIN MONITOR.
174 (SEE SECTION 9. FOR DETAILS)
175
176
177
178 [3.0] LOADING PROCEDURE
179
180 USE STANDARD PROCEDURE FOR PDP-11 ABSOLUTE BINARY FORMATTED TAPES.
181
182
183
184 [4.0] STARTING PROCEDURE
185
186 [4.1] SWITCH SETTINGS
187
188 SOFTWARE SWITCH REGISTER = LOCATION 176
189
190 BIT15(100000) HALT ON ERROR
191
192 BIT14(040000) LOOP ON TEST DEFINED BY SWITCH REGISTER BITS <310>
193
194 BIT13(020000) INHIBIT ERROR PRINTOUTS
195
196 BIT12(010000) ENABLE MEMORY MANAGEMENT (TESTING ABOVE 28K, 30K SYSTEM DOES NOT
197 NEED KT SUPPORT)
198
199 BIT11(004000) ENABLE PARITY MODULES.
200 !"PAR" WILL BE TYPED
201
202 BIT10(002000) HALT AFTER EACH SUBTEST
203 !PRESS CONTINUE TO DO NEXT SUBTEST
204
205 BIT09(001000) INHIBIT PROGRAM RELOCATION
206 !IF SET LOCATIONS 430-7776 WILL NOT BE
207 !TESTED.
208
209 BIT08(000400) TYPE FIRST FAILING BIT IN EACH 4K BANK ONLY.
210 !THE TOTAL ERROR COUNT (UP TO 377) WILL
211 !BE SAVED IN THE ERROR HISTORY.
212
213 BIT07(000200) ENABLE LONG GALLOPING TEST.
214 !"GLP" WILL BE TYPED.
215 !CAUTION! INCREASES TEST TIME BY FACTOR OF 25.

216
217 BIT06(000100) INHIBIT MEMORY SIZING.
218 !THE MEMORY LIMITS MUST BE SETUP IN THE FOLLOWING LOCATIONS:
219 (VALUES TO TEST 0-8K ARE SHOWN)
220 (LOWTWO=LOCATION 324)
221
222 LOWTWO: 0 ;STORE BITS 17:16 OF LOW TEST ADDRESS
223 LOWADD: 0 ;STORE REST OF LOW TEST ADDRESS
224 ;DO NOT ATTEMPT TO SET THE LOWER LIMIT
225 ;AT OR ABOVE 160000 ON A 30K LSI SYSTEM.
226 ;THE PROGRAM WILL ASSUME MEMORY MANAGEMENT
227 ;MUST BE USED.
228 HIGHTWO: 0 ;STORE BITS 17:16 OF HIGH TEST ADDRESS
229 HIGHADD: 37776 ;STORE REST OF HIGH TEST ADDRESS
230
231 BIT05(000040) INHIBIT "PASS#XX" PRINTOUTS
232
233 BIT04(000020) A. INHIBIT EPROF HISTORY PRINTOUTS. THE
234 ERROR HISTORY CAN STILL BE OBTAINED
235 BY TYPING CONTROL-C.
236 B. INHIBIT PRINTOUTS "PAR","GLP","TST13 BNK XX".
237
238 BIT03-BIT00 NUMBER OF TEST (0-13) TO RUN FIRST.
239 INORMALLY USED WITH BIT14 (LOOP ON TEST)
240
241
242
243 [4.2] CONTROL-C OPTION
244
245 CONTROL C [^C] AFTER COMPLETION OF THE CURRENT TEST.
246 THE EPROF HISTORY (SEE SEC. 6.3) WILL BE
247 TYPED. THE PROGRAM WILL HALT IN LOWER MEMORY.
248 PRESSING CONTINUE WILL RESTART THE DIAGNOSTIC.
249
250 [4.3] STARTING ADDRESS= 200
251 RESTART ADDRESS = 250 OR 200
252
253 RESTART AT 200 CLEARS PASS COUNT (\$PASS) AND PRINTS "CZKMAF" TITLE.
254
255
256
257
258
259 [4.4] PROGRAM AND/OR OPERATOR ACTION
260
261 1) LOAD PROGRAM INTO MEMORY USING ABSOLUTE LOADER.
262 2) SET OPTIONS (SEE SEC. 4.1)
263 3) START THE PROGRAM AT 200
264 4) THE FOLLOWING IS AN EXAMPLE WITH EXPLANATIONS
265 OF THE PRINTOUTS EXPECTED.
266
267 "XXXXX-YYYYY" ;ADDRESSES OF TEST BOUNDARIES.
268
269 "PAR" ;IF PARITY OPTION SELECTED
270
271 "GLP" ;IF LONG GALLOPING OPTION SELECTED.

272 ;PRINTED AS TST11 IS ENTERED.
 273
 274 "TST13 BNK 00" ;ENTERING BANK 00 IN TEST 13.
 275 "TST13 BNK 01" ;AND BANK 1...
 276 ETC... ;UNTIL ALL BANKS (UP TO 7) HAVE BEEN TESTED.
 277
 278 "REL" ;THE DIAGNOSTIC RELOCATES TO HIGHEST
 279 ;LOCATIONS UNDER TEST AND RUNS TST0-TST13 AGAIN.
 280
 281 "TST13 BNK 00" ;TESTING BANK 00 IN TEST 13 (RELOCATED STATE.)
 282 ;NOTE-ONLY BANK 00 IS TESTED IN THE RELOCATED STATE.
 283
 284 "PASS#XX" ;WHERE "XX" IS THE PASS NO.
 285
 286
 287 ADDITIONAL PRINTOUTS
 288 "NO PAR" ;PRINTED IF PARITY SELECTED BUT NOT AVAILABLE.
 289
 290 "NO KT" ;PRINTED IF SWR BIT 12 IS SET AND NO MEMORY
 291 ;MANAGEMENT AVAILABLE.
 292
 293
 294

295 4.5 LONG GALLOP OPTION

296
 297 NORMAL WORST CASE SR SETTING = 0000. FOP LONG GALLOP
 298 SR = 200. LONG GALLOP OPTION SHOULD ONLY BE USED IF AN
 299 MOS MEMORY PROBLEM IS SUSPECTED AND NO OTHER SUBTESTS
 300 WILL FAIL. THE TEST TIME IS INCREASED 25 TIMES.
 301
 302
 303

304 [5.0] PROGRAM HALTS (NORMAL+ ERROR)

305
 306 THIS IS A LIST OF EXPECTED HALTS. IF THE TEST HALTS
 307 IN A LOCATION NOT IN THIS LIST AND IT IS LESS THAN 776, IT
 308 MAY BE DUE TO A DEVICE INTERRUPTING.
 309 NOTE THE HALT AT END OF SUBTEST AND HALT ON ERROR HALT LOCATIONS
 310 MAY BE RELOCATED. THE ACTUAL LOCATIONS THEY ARE IN CAN BE FOUND
 311 BY SUBTRACTING 500 FROM THE HALT PC AND ADDING THIS DIFFERENCE TO THE
 312 CONTENTS OF SAVR6 [LOC. 350].
 313

| PC | REASON | RECOVERY |
|------|---------------------------------|--|
| -- | ----- | ----- |
| 112 | TRAP TO LOC. 4 | EXAMINE R6, IT CONTAINS THE POINTER TO THE PC WHERE THE TRAP OCCURRED. |
| --- | POWER FAIL | POWER UP WILL RECOVER IF IN CORE MEMORY. IF NOT CORE OPERATION IS UNDEFINED. |
| 1714 | HALT AT END OF TEST SWITCH SET. | PRESS CONTINUE TO GO TO NEXT SUBTEST. |

328 6156 HALT ON ERROR PRESS CONTINUE.
329 SWITCH SET.
330
331 6240 CONTROL-C TYPED PRESS CONTINUE TO RE-
332 OR FATAL ERROR START TEST.
333 OCCURRED
334

335
336 [6.0] ERRORS

337
338 [6.1] ERROR MESSAGE FORMAT

339
340 THE ERROR PRINTOUT CONSISTS OF 6 OCTAL WORDS IN THE FOLLOWING
341 FORMAT:

342
343 "LOCATION GOOD BAD PC ERROR PASFLG"

344
345
346 "ADR ERR" WILL BE PRINTED PRIOR IF AN ADDRESSING ERROR IS SUSPECTED.
347 "PAR ERR" WILL BE PRINTED PRIOR IF A PARITY ERROR TRAP OCCURRED
348 !CAUTION! IF PARITY ERROR THE GOOD DATA PRINTOUT IS THE
349 PARITY MODULE UNIBUS ADDRESS THAT FAILED.

350

351

352 WHERE:

353

354 LOCATION= FAILING MEMORY LOCATION
355 GOOD = GOOD DATA [DATA THAT WAS EXPECTED]
356 BAD = BAD DATA [DATA THAT WAS FOUND]
357 PC = PROGRAM COUNTER AT ERROR CALL.
358 ERROR = FAILING ERROR NO. (SEE SEC 6.2 - ERROR DICTIONARY)
359 PASFLG = CONTENTS OF LOCATION PASFIG. THIS MAY NOT BE RELEVANT.
360 (SEE SEC. 6.2-ERROR DICTIONARY)

361

362

363 !THE TEST WILL CONTINUE AFTER THE ERROR PRINTOUT.
364 !"NO KT" WILL BE TYPED IF TESTING ABOVE 2RK SELECTED AND NO MEMORY
365 MANAGEMENT IS FOUND.(30K SYSTEM DOES NOT NEED KT SUPPORT)

366

367 !"NO PAR" WILL BE TYPED IF PARITY OPTION SELECTED
368 !AND NO PARITY MODULES WERE FOUND.

369

370

371

372 "ERR #XXXXXX" WILL BE TYPED WHERE "XXXXXX" IS
373 THE ERROR NUMBER, THE DIAGNOSTIC WILL USUALLY HALT ON THIS TYPE
374 OF ERROR. SEE SEC. 6.2 -ERROR DICTIONARY - FOR DESCRIPTIONS
375 OF THE ERROR.

376

377

378

379 (APT MODE ERRORS)

380

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

384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439

[6.2] ERROR DICTIONARY

\$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.
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. IF PARITY SELECTED IT HAS A VALUE = 376 ,ELSE=377
"SWAPPED BAKPAT" = 77000 IF PARITY SELECTED, ELSE=77400

.ENDR

;ERR # 0 ;[BUSER] BUS ERROR TRAP TO LOC. 4 OCCURRED
; THIS ERROR IS NOT PRINTED AND IS FOR "APT" USE.

;ERR # 1 ;[TSTTRP]FATAL DATA ERROR
;LOCATIONS 0000-430 FAILED 1'S + 0'S TEST.
;R0 = GOOD DATA
;R1 = ADDRESS OF FAILING LOCATION.

;ERR # 2 ;[APTSIZ] APT FATAL ERROR
;APT MEMORY TABLES NOT SETUP CORRECTLY.
;CHECK LOCATIONS \$MAMS1 [430] TO \$MADR4[446]
; FOR CORRECT MEMORY SIZE DATA.

;ERR # 3 ;[TSTSIZ] OPERATOR FATAL ERROR
;SELECTED MEMORY SIZE GREATER THAN 28K
;(30K SYSTEM DOES NOT NEED KT SUPPORT), BUT
;SR BIT12 (10000) NOT SET.
;SET BIT12 AND RESTART AT 200.

;ERR # 4 ;[TSTSIZ] OPERATOR FATAL ERROR
;LOWEST SELECTED TEST LIMIT IS HIGHER THAN
;HIGHEST TEST LIMIT. SET LOCATIONS "LOWTWO"[322]
;TO "HIGHADD" [330] CORRECTLY AND RESTART
;AT 200.

;ERR # 5 ;[TST0] TEST SEQUENCE ERROR
;TST0 HAS BEEN ENTERED OUT OF SEQUENCE
;TESTN SHOULD = 00
;THE DIAGNOSTIC HAS BEEN CORRUPTED.
;IF POSSIBLE SELECT ANOTHER 4K BANK
;BANK 0 AND RERUN THE TEST ON THE FAILING MEMORY.

;ERR # 6 ;[TST0] DUAL ADDRESSING ERROR
;FOR THIS ERROR THE GOOD DATA PRINTED IS AN
;ADDRESS. THIS IS THE ADDRESS SELECTED WHEN
;THE SAME DATA WAS WRITTEN INTO THE FAILING


```
440 ;LOCATION. CHECK BANK SELECT CIRCUITRY
441
442 ;ERR # 7 ;[TST0] ADDRESS AND DATA ERROR
443 ;IDENTICAL TO PREVIOUS ERROR EXCEPT THE DATA
444 ;WRITTEN INTO THE FAILING LOCATION WAS IN
445 ;ERROR ALSO.
446
447 ;ERR # 10 ;[TST0] DATA ERROR
448 ;IF BAD DATA = 0000 COULD BE AN ADDRESSING
449 ;ERROR , ELSE COMPARE GOOD AND BAD DATA FOR FAILING BITS.
450
451 ;ERR # 11 ;[TST0] ADDRESSING ERROR
452 ;THE FAILING ADDRESS RESPONDED BUT IS NON-
453 ;EXISTENT. MAY BE A DUAL ADDRESSING PROBLEM.
454
455 ;ERR # 12 ;[TST1] TEST SEQUENCE ERROR
456 ;#TESTN [404] SHOULD = 01
457 ; THE DIAGNOSTIC HAS BEEN CORRUPTED.
458
459 ;ERR # 13 ;[TST1] DATA ERROR
460 ;COMPARE GOOD AND BAD PRINTED DATA, FAILING
461 ;DATA BITS MAY SHORTED OR SWAPPED.
462
463 ;ERR # 14 ;[TST2] TEST SEQUENCE ERROR
464 ;#TESTN [404] SHOULD = 02
465 ; THE DIAGNOSTIC HAS BEEN CORRUPTED.
466
467 ;ERR # 15 ;[TST2] ADDRESS OR DATA ERROR
468 ;IF "ADR ERR" NOT PRINTED THEN THE BYTE SELECT
469 ;CIRCUITRY PROBABLY FAILED.
470
471 ;ERR # 16 ;[TST3] TEST SEQUENCE ERROR
472 ;#TESTN [404] SHOULD = 03
473 ; THE DIAGNOSTIC HAS BEEN CORRUPTED.
474
475 ;ERR # 17 ;[TST3] DUAL ADDRESSING ERROR
476 ;DUAL ADDRESSING PROBLEM FOR BITS THAT DIFFER
477 ;IN GOOD AND BAD DATA PRINTOUT.
478
479 ;ERR # 20 ;[TST3] DUAL ADDRESSING ERROR
480 ;FOR THIS ERROR THE DATA PRINTED IS AN ADDRESS.
481 ;THIS IS THE ADDRESS THAT WAS SELECTED WHEN THE
482 ;SAME DATA WAS WRITTEN INTO THE FAILING LOCATION.
483
484 ;ERR # 21 ;[TST3] DUAL ADDRESSING ERROR
485 ;SAME AS ERROR #20 EXCEPT DIFFERENT DATA
486 ;(SWAPPED BAKPAT) WAS WRITTEN.
487
488 ;ERR # 22 ;[TST4] TEST SEQUENCE ERROR
489 ;#TESTN [404] SHOULD = 04.
490 ; THE DIAGNOSTIC HAS BEEN CORRUPTED.
491
492 ;ERR # 23 ;[TST4] DUAL ADDRESSING ERROR
493 ;IF PASFLG = 0 THEN THE FAILING LOCATION
494 ;AND FAILING DATA ARE DUAL ADDRESSES.
495
```

```
496 ;ERR # 24 ;[TST5] TEST SEQUENCE ERROR
497 ;$TESTN [404] SHOULD = 05
498 ; THE DIAGNOSTIC HAS BEEN CORRUPTED.
499
500 ;ERR # 25 ;[TST5] DATA ERROR
501 ;DATA WRITE OR READ ERROR.
502 ;ERR # 26 ;[TST5] MARCHING 1'S AND 0'S DATA ERROR
503 ;IF PASFLG=0 FAILED MARCHING 1'S + 0'S IN
504 ; MAX TO MIN DIRECTION.
505 ;IF PASFLG=1 FAILED MARCHING 1'S + 0'S IN
506 ; MIN TO MAX DIRECTION.
507 ;IF PASFLG=3 FAILED MARCHING 0'S + 1'S IN
508 ; MAX TO MIN DIRECTION.
509
510 ;ERR # 27 ;[TST5] MARCHING 1'S AND 0'S DATA ERROR
511 ;IDENTICAL TO PREVIOUS ERROR EXCEPT THE DATA IS
512 ;CHECKED IMMEDIATELY AFTER BEING WRITTEN.
513
514 ;ERR # 30 ;[TST6] TEST SEQUENCE ERROR
515 ;$TESTN SHOULD = 06
516 ;THE DIAGNOSTIC HAS BEEN CORRUPTED.
517
518 ;ERR # 31 ;[TST6] VOLATILITY/REFRESH TEST ERROR
519 ;IF PASFLG=0 BAKPAT WRITE OR READ ERROR.
520 ;IF PASFLG=1 THE FAILING LOCATION CHANGED WHILE
521 ; ANOTHER LOCATIONS WAS WRITTEN FOR
522 ; 2 MS. THE OTHER LOCATION IS SAVED
523 ; IN SAVLOC (352)
524 ;IF PASFLG=2 SWAPPED BAKPAT (77400 OR 77000)
525 ; WRITE OR READ ERROR.
526 ;IF PASFLG=3 SAME AS IF PASFLG=2 EXCEPT
527 ; THE DATA IS SWAPPED BAKPAT.
528
529 ;ERR # 32 ;[TST7] TEST SEQUENCE ERROR
530 ;$TESTN SHOULD = 07
531 ;THE DIAGNOSTIC HAS BEEN CORRUPTED.
532
533 ;ERR # 33 ;[TST7] SHIFTING DIAGONAL DATA ERROR
534 ;IF PASFLG=0 BAKPAT WRITE OR READ ERROR.
535 ;IF PASFLG=1 BAKPAT READ CHECK ERROR
536 ;IF PASFLG GREATER THAN 1 BUT EVEN VALUE THEN:
537 ; THE FAILING LOCATION COULD NOT BE WRITTEN INTO.
538 ;IF PASFLG GREATER THAN 1 BUT ODD VALUE THEN:
539 ; THE FAILING LOCATION WAS WRITTEN CORRECTLY
540 ; BUT LOST THE DATA.
541
542 ;ERR # 34 ;[TST10] TEST SEQUENCE ERROR
543 ;$TESTN SHOULD = 10
544 ; THE DIAGNOSTIC HAS BEEN CORRUPTED.
545
546 ;ERR # 35 ;[TST10] BAKPAT DATA ERROR
547 ;BAKPAT WRITE OR READ ERROR INTO THE FAILING LOCATION.
548
549 ;ERR # 36 ;[TST10] READ RECOVERY DATA ERROR
550 ; THIS ERROR CAN BE REPORTED BY TST10 AND TST11.
551 ;(THEY SHARE CODE). SEE $TESTN [404] FOR WHICH TEST FAILED.
```

```
552 ;FOR BOTH TESTS COMPARE THE GOOD AND BAD DATA AT THE FAILING
553 ;LOCATION TO SEE WHICH BITS FAILED.
554
555 ;ERR # 37 ;[TST10] READ RECOVERY DATA ERROR
556 ;IDENTICAL TO THE PREVIOUS ERROR EXCEPT SWAPPED BAKPAT IS
557 ;USED AS WRITE AND READ DATA.
558
559 ;ERR # 40 ;[TST11] TEST SEQUENCE ERROR
560 ;$TESTN SHOULD = 11
561 ; THE DIAGNOSTIC HAS BEEN CORRUPTED.
562
563 ;ERR # 41 ;[TST12] TEST SEQUENCE ERROR
564 ;$TESTN SHOULD = 12
565 ; THE DIAGNOSTIC HAS BEEN CORRUPTED.
566
567 ;ERR # 42 ;[TST12] WORST CASE CORE TEST DATA ERROR
568 ;IF PASFLG=1 COMPARE GOOD AND BAD DATA FOR FAILING BITS.
569 ;IF PASFLG=2 THE FAILING LOCATION WAS WRITTEN AND READ
570 ; WITH GOOD DATA,BUT FAILED READ CHECK
571 ; READING IN THE MIN. TO MAX DIRECTION.
572 ;IF PASFLG=3 SAME CONDITIONS AS PASFLG=2 EXCEPT FAILED
573 ; DOING THE READ CHECK FROM MAX TO MIN DIRECTION.
574
575 ;ERR # 43 ;[TST12] WORST CASE CORE TEST DATA ERROR
576 ; IDENTICAL TO PREVIOUS ERROR EXCEPT THE DATA WRITTEN
577 ;AND READ IS COMPLEMENTED.
578
579 ;ERR # 44 ;[TST13] TEST SEQUENCE ERROR
580 ;$TESTN SHOULD = 13
581 ; THE DIAGNOSTIC HAS BEEN CORRUPTED.
582
583 ;ERR # 45 ;[TST13] WRITE RECOVERY TEST DATA ERROR
584 ;IF PASFLG=0 COMPARE GOOD AND BAD DATA FOR FAILING BITS.
585 ;IF PASFLG=77400 DATA ERROR FOUND WHILE DOING A SECOND READ CHECK.
586 ;IF PASFLG=77402 DATA ERROR FOUND IN FAILING LOCATION AFTER
587 ; SMALL TEST PROGRAM RUN IN FAILING BANK.
588
589 ;ERR # 46 ;[TST13] WRITE RECOVERY TEST DATA ERROR
590 ; DATA ERROR FOUND JUST BEFORE THE SMALL TEST
591 ;WAS TO BE RUN IN THE FAILING BANK. TO AVOID "BLOWING" UP
592 ;WHEN THE SMALL TEST IS RUN TST13 IS ABORTED.
593
594 ;ERR # 47 ;[TST13] WRITE RECOVERY TEST DATA ERROR
595 ; IDENTICAL TO ERROR #XXX EXCEPT THE DATA WRITTEN
596 ;AND READ IS DIFFERENT.(177667).
597 ;177667 IS THE COMPLEMENT OF "JMP (R0)" (110) WHICH IS
598 ;THE ESCAPE FROM THE SMALL TEST PROGRAM RUN IN THE BANK
599 ;UNDER TEST.
600
601 ;ERR # 50 ;[PARERR] PARITY TRAP ERROR
602 ; PARITY TRAP TO 114 OCCURRED.
603 ;FOR THIS ERROR PRINTOUT THE "GOOD DATA" IS ACTUALLY
604 ;THE FAILING PARITY MODULE UNIBUS ADDRESS.
605 ; SAVLOC [352] CONTAINS THE PC WHERE THE TRAP OCCURRED.
606
607 ;ERR # 51 ;[PARITY] PARITY TRAP FATAL ERROR
```

608 ; A PARITY TRAP TO 114 OCCURRED, BUT NO PARITY MODULES COULD BE FOUND
609 ;WITH AN ERROR BIT (BIT15) SET.
610
611 ;EPR # 52 ;(NOMM) OPERATOR FATAL ERROR
612 ; TESTING ABOVE 28K WAS SELECTED, BUT NO MEMORY MANAGEMENT
613 ;OPTION WAS FOUND.(30K SYSTEM DOES NOT NEED MT)
614 ; RESET SWITCH OPTIONS AND RESTART AT 200.
615
616 ;EPR # 53 ;(PARITY) OPERATOR FATAL ERROR
617 ; PARITY TESTING WAS SELECTED BUT NO PARITY MODULES
618 ;WERE FOUND.
619 ; RESET SWITCH OPTIONS AND START AT 200.
620
621 .REPT 0
622
623
624 [6.3] ERROR HISTORY
625
626 LOCATIONS IN MEMORY ARE SET ASIDE TO COLLECT A HISTORY
627 OF THE FAILING BITS IN A PARTICULAR MEMORY BANK. THIS
628 DATA IS COLLECTED FOR EVERY ERROR REGARDLESS OF SWITCH
629 SETTINGS.
630
631 NORMALLY THE DATA IS OUTPUT AT THE END OF TESTING, BUT
632 IF CONTROL-C IS TYPED IT IS OUTPUT AT THE END OF THE
633 CURRENT TEST.
634
635 THE ERROR HISTORY IS INTENDED TO HIGHLIGHT IF THE ERRORS
636 ARE DUE TO 1 BIT FAILING OR ONLY ADDRESS ERRORS.
637
638
639
640 ERROR HISTORY FORMAT:
641
642
643 ERROR BANK COUNT
644 -----
645
646
647 WHERE:
648
649 ERROR = BIT THAT FAILED (NUMBER OF THE FAILING BIT IN DECIMAL I.E.
650 0-15 WILL BE TYPED OUT OR THE WORDS "ADR ERR" OR "PAR ERR" WILL
651 BE TYPED OUT IF ADDRESS ERROR OR PARITY ERROR WAS SEEN
652 IN THE SPECIFIC BANK OF MEMORY
653 BANK = 4K MEMORY BANK IN WHICH THIS FAILURE WAS SEEN
654 A 0 FOR 0 TO 4K, A 1 FOR 4 TO 8K AND SO ON
655 COUNT = NUMBER OF TIMES THIS MEMORY BANK FAILED.
656 (377 IS MAXIMUM FAILURE COUNT RECORDED.)
657 [6.4] ERROR RECOVERY
658
659 IF THE PROGRAM IS HALTED AFTER REPORTING AN ERROR IT CAN EITHER
660 BE CONTINUED OR RESTARTED AT 200 OR 250 (SEE SEC 4.2). HOWEVER FOR
661 CPU'S THAT DESTROY CONTENTS OF REGISTERS AFTER COMING TO A HALT
662 THE PROGRAM SHOULD ONLY BE RESTARTED.
663

664
 665 [7.0] RESTRICTIONS
 666

667 MEMORY UNDER TEST SHOULD BE CONTIGUOUS. FOR SYSTEMS HAVING NON-
 668 CONTIGUOUS MEMORY THE MEMORY BOUNDARIES SHOULD BE DEFINED BY THE
 669 OPERATOR. (CONTIGUOUS MEMORY IS DEFINED AS A MEMORY THAT CAN BE
 670 BOTH READ AND WRITTEN IN CONSECUTIVE LOCATIONS.)
 671

672
 673
 674
 675 [8.0] MISCELLANEOUS
 676

677
 678 [8.1] ADDRESS/BANK RANGES IN OCTAL AND DECIMAL
 679

680 THIS REFERENCE TABLE CROSS REFERENCES THE MEMORY BANK NO.S,
 681 THE RANGE AND THE PAR USED WHEN MEMORY MANAGEMENT IS ENABLED.
 682 IT IS ALSO USEFUL TO SHOW STARTING ADDRESSES IN A PAR-
 683 TICULAR 4K BANK.
 684

| BANK NO. | DECIMAL RANGE | OCTAL RANGE | [PAGE ADDRESS REGISTER] USED/CONTENT | UNIBUS ADDRESS |
|----------|---------------|---------------|--------------------------------------|----------------|
| 685 | | | | |
| 686 | | | | |
| 687 | | | | |
| 688 | 0 - 4K | 000000-017776 | 0 0000 | 772340 |
| 689 | 4K - 8K | 020000-037776 | NOT USED | |
| 690 | 8K-12K | 040000-057776 | NOT USED | |
| 691 | 12K-16K | 060000-077776 | NOT USED | |
| 692 | | | | |
| 693 | 4 16K-20K | 100000-117776 | NOT USED | |
| 694 | 5 20K-24K | 120000-137776 | NOT USED | |
| 695 | 6 24K-28K | 140000-157776 | NOT USED | |
| 696 | 7 28K-32K | 160000-177776 | NOT USED ON 30K (LSI-11) SYSTEMS | |
| 697 | | | 1 1600 | 772342 |
| 698 | | | | |
| 699 | 8 32K-36K | 200000-217776 | 2 2000 | 772344 |
| 700 | 9 36K-40K | 220000-237776 | 3 2200 | 772346 |
| 701 | 10 40K-44K | 240000-257776 | 4 2400 | 772350 |
| 702 | 11 44K-48K | 260000-277776 | 5 2600 | 772352 |
| 703 | | | | |
| 704 | 12 48K-52K | 300000-317776 | 6 3000 | 772354 |
| 705 | 13 52K-56K | 320000-337776 | 1 3200 | |
| 706 | 14 56K-60K | 340000-357776 | 2 3400 | |
| 707 | 15 60K-64K | 360000-377776 | 3 3600 | |
| 708 | | | | |
| 709 | 16 64K-68K | 400000-417776 | 4 4000 | |
| 710 | 17 68K-72K | 420000-437776 | 5 4200 | |
| 711 | 18 72K-76K | 440000-457776 | 6 4400 | |
| 712 | 19 76K-80K | 460000-477776 | 1 4600 | |
| 713 | 20 80K-84K | 500000-517776 | 2 5000 | |
| 714 | 21 84K-88K | 520000-537776 | 3 5200 | |
| 715 | 22 88K-92K | 540000-557776 | 4 5400 | |
| 716 | 23 92K-96K | 560000-577776 | 5 5600 | |
| 717 | 24 96K-100K | 600000-617776 | 6 6000 | |
| 718 | | | | |
| 719 | 25 100K-104K | 620000-637776 | 1 6200 | |

| | | | | | |
|-----|-------|--|---------------|---|------|
| 720 | 26 | 104K-108K | 640000-657776 | 2 | 6400 |
| 721 | 27 | 108K-112K | 660000-677776 | 3 | 6600 |
| 722 | 28 | 112K-116K | 700000-717776 | 4 | 7000 |
| 723 | | | | | |
| 724 | 29 | 116K-120K | 720000-737776 | 5 | 7200 |
| 725 | 30 | 120K-124K | 740000-757776 | 6 | 7400 |
| 726 | 31 | 124K-128K | 760000-777776 | 7 | 7600 |
| 727 | | | | | |
| 728 | | | | | |
| 729 | | | | | |
| 730 | | | | | |
| 731 | | | | | |
| 732 | | | | | |
| 733 | | | | | |
| 734 | | | | | |
| 735 | | | | | |
| 736 | | | | | |
| 737 | | | | | |
| 738 | [8.2] | EXECUTION TIME | | | |
| 739 | | | | | |
| 740 | | HERE ARE SOME TYPICAL EXECUTION TIMES. | | | |
| 741 | | | | | |
| 742 | | LSI-11 AND 4K: = 100 SECS. | | | |
| 743 | | LSI-11 AND 8K: = 5 MINUTES. | | | |
| 744 | | | | | |
| 745 | | | | | |
| 746 | | | | | |
| 747 | [8.2] | PASS COUNT AND TEST NO. LOCATIONS | | | |
| 748 | | | | | |
| 749 | | \$PASS [406] = PASS COUNT - CLEARED BY START AT 200. | | | |
| 750 | | | | | |
| 751 | | \$TESTN [404] = CURRENT TEST NO. AND RELOCATION, PARITY FLAGS. | | | |
| 752 | | WHERE: | | | |
| 753 | | LOW BYTE = TEST NO. | | | |
| 754 | | IF BIT15 = 1 TEST IS RELOCATED | | | |
| 755 | | IF BIT13 = 1 PARITY UNDER TEST. | | | |
| 756 | | | | | |
| 757 | | | | | |
| 758 | [8.4] | STACK POINTER | | | |
| 759 | | | | | |
| 760 | | THE STACK STARTS AT 500 WHEN THE PROGRAM IS NOT RELOCATED. | | | |
| 761 | | SAVR6[350] CONTAINS THE STACK STARTING VALUE WHEN THE DIAGNOSTIC | | | |
| 762 | | IS RELOCATED. | | | |
| 763 | | SAVR6 ALSO CONTAINS THE STARTING ADDRESS OF THE PROGRAM WHEN | | | |
| 764 | | IT IS RELOCATED. | | | |
| 765 | | | | | |
| 766 | [8.5] | POWER FAIL | | | |
| 767 | | | | | |
| 768 | | THE DIAGNOSTIC CAN BE POWER FAILED WITH NO ERRORS. TO USE, | | | |
| 769 | | START THE TEST AS USUAL AND POWER DOWN THEN UP AT ANY TIME. | | | |
| 770 | | THE PROGRAM SHOULD TYPE "P" AND CONTINUE TO RUN FROM TEST 0 | | | |
| 771 | | IN THE SAME STATE [I.E. STATE OF RELOCATION] AS IT WAS BEFORE | | | |
| 772 | | THE POWER WAS INTERRUPTED, HOWEVER IF THE DIAGNOSTIC WAS IN | | | |
| 773 | | A MEMORY THAT CAN NOT HOLD DATA WITH THE POWER DOWN THEN THE | | | |
| 774 | | PROGRAM WILL NOT RECOVER FROM POWER FAIL AND ON POWER-UP | | | |
| 775 | | OPERATION IS UNDEFINED. | | | |

NOTES:

1. THE PAR (PAGE ADDRESS REGISTER) CONTENTS ARE SHOWN IN A TEST THAT SELF SIZES. IF THE LIMITS OF TESTING ARE SET BY THE OPERATOR AND IF THE BANK IS ABOVE 28K PAR NO. 1 WILL BE SET TO THE BEGINNING PAGE. FOR EXAMPLE IF THE TESTING WAS TO BEGIN WITH BANK 8 PAR NO. 1 WOULD EQUAL 2000, PAR 2 WOULD EQUAL 2200 ETC.

772354

776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831

[9.0] PROGRAM DESCRIPTION

[9.1] NARRATIVE FLOW CHART

THE TEST IS LOADED INTO LOCATIONS 0000 - 7744 BUT EXPANDS DEPENDING ON HOW MUCH MEMORY IS UNDER TEST. SEE STEP 6. BELOW FOR A DETAILED EXPLANATION.

THE FOLLOWING NARRATIVE FLOW CHART DESCRIBES MAJOR PROGRAM OPERATION. FOR THE PERSON WHO NEEDS DETAIL THE TAG ASSOCIATED WITH THE OPERATION IS GIVEN IN BRACKETS.

FOR THIS DISCUSSION SWITCH SETTINGS ARE IGNORED AND EVERYTHING IS ASSUMED ENABLED.

1. [START] PRINT "CZKMAF" TITLE
2. [TSTRP] SAVE DATA FROM LOCATIONS 0-376 INTO 7744-10314.
3. [TSTRP] TEST LOCATIONS 0-376 BY WRITING AND READING 1'S AND 0'S. NOTE THIS IS THE ONLY EXPLICIT TESTING OF THESE LOCATIONS.
4. [SLFSIZ] SIZE MEMORY BY WRITING INTO SUCCEEDING MEMORY LOCATIONS UNTIL TIMEOUT TRAP TO 4 OCCURS, OR 30K BOUNDARY REACHED. ENABLE MEMORY MANAGEMENT AND SIZE MEMORY ABOVE 28K. NOTE: IF UNDER XXDP CHAIN MODE IN 30K SYSTEM, SYSTEM IS SIZED TO 28K.
5. [TYPSIZ] TYPE MEMORY TEST LIMITS.
6. [SETSTK] SPACE IS SAVED AT THE END OF THE TEST FOR AN ERROR HISTORY. FOR EACH 4K BANK 18 BYTES ARE SAVED IN THE FOLLOWING FORMAT:

```
!ADR ERR!PAR ERR!  
!BIT14 !BIT15 !  
!BIT12 !BIT13 !  
!BIT10 !BIT11 !  
!BIT08 !BIT09 !  
!BIT06 !BIT07 !  
!BIT04 !BIT05 !  
!BIT02 !BIT03 !  
!BIT00 !BIT01 !
```

IF GREATER THAN 4K UNDER TEST THE ABSOLUTE LOADER (300 ADDRESSES) IS APPENDED. IF GREATER THAN 4K AND UNDER XXDP CHAIN MODE 5376 (OCTAL) ADDRESSES ARE APPENDED TO THE TEST. THIS SAVES THE XXDP

832 MONITOR, AND ALLOWS THE LOCATIONS OCCUPIED BY XXDP
833 TO BE TESTED.
834
835 7. [CLRMEM] CALL "PARITY" ROUTINE AND IF SELECTED,
836 ENABLE ALL PARITY MODULES. "PARMAP" [LOC. 352]
837 CONTAINS A MAP OF PARITY MODULES FOUND. IF
838 MODULE 172336 BIT 15 IS SET, IF #172334 FOUND BIT 14
839 IS SET ETC.,
840
841 8. [CLRMEM] CLEAR MEMORY CURRENTLY UNDER TEST
842
843 9. [CONT] DISPATCH TO TST0
844
845 10. [TST0] EXECUTE TEST 0. SEE SECTION 10 FOR TEST
846 DESCRIPTIONS.
847
848 11. [TSTSCP] COMES HERE AFTER EACH TEST AND IF
849 CNTRL-C TYPED THEN GO TO ERROR HISTORY PRINTOUT.
850 IF SR=2000 THEN HALT
851 IF SR=4000 THEN LOOP ON TEST DEFINED BY <3:0>
852 ELSE CONTINUE TO NEXT TEST.
853
854 12. [TST1-TST12] EXECUTE TST1-TST12 EACH TIME
855 GOING TO STEP 9.
856
857 13. [TST13] TEST 13 IS DIFFERENT FROM TESTS 0-12,
858 BECAUSE IT IS A SMALL PROGRAM ACTUALLY RUNNING
859 IN THE MEMORY UNDER TEST. BEFORE THIS SMALL
860 PROGRAM IS STARTED "TST13 BNK XX" IS TYPED.
861 THIS IS DONE IN CASE THE PROGRAM FAILS. THE
862 USER CAN THEN AT LEAST TELL WHICH BANK OF MEMORY
863 FAILED.
864
865 14. [RELOC] THE PROGRAM RELOCATES TO HIGH MEMORY
866 TO TEST THE LOCATIONS IT OCCUPIES. (430-ENDPRG).
867 WHERE "ENDPRG" IS THE CONTENTS OF ENDSTK[306].
868 I.E THE LAST PROGRAM ADDRESS. NOTE "REL" IS
869 PRINTED JUST PRIOR TO THE ACTUAL RELOCATION.
870
871 15. TESTS 0-13 ARE RUN AS DESCRIBED ABOVE EXCEPT
872 ONLY BANK 0 LOCATIONS 430-ENDPRG ARE TESTED.
873
874 16. [RELOER] RELOCATE THE PROGRAM BACK TO LOWER
875 MEMORY.
876
877 17. [LOWER] IF CONTROL-C TYPED GO PRINT ERROR
878 HISTORY.
879
880 18. [TSTMM] IF MEMORY MANAGEMENT SELECTED AND AVAILABLE,
881 RUN TESTS 0-13 ON THE FIRST 24K SLICE ABOVE 28K.
882
883 19. [CONTMM] CALL "UPMM" TO UPDATE MEMORY MANAGEMENT
884 PAR REGISTERS TO POINT TO THE NEXT 24K SLICE OF
885 UPPER MEMORY.
886
887 20. [MAXADR] REPEAT STEPS 18 + 19 UNTIL ALL

888 MEMORY ABOVE 28K IS TESTED.
889
890 21. [ENDPAS] PRINT ERROR HISTORY OF FAILING BITS
891
892 22. [#EOP] DISABLE PARITY MODULES.
893 PRINT "PASS#XX"
894
895
896
897 [9.2] TEST TITLES
898
899 SEE THE TEST HEADINGS IN THE LISTING FOR DETAILS ON EACH TEST.
900
901 TEST 0: TEST FOR PROPER BANK SELECTION
902 TEST 1: CHECK DATI/DATO LINES
903 TEST 2: TEST MEMORY FOR HOLDING DATA AND BYTE SELECTION
904 TEST 3: DUAL ADDRESS TEST A
905 TEST 4: DUAL ADDRESS TEST B
906 TEST 5: MARCHING 1'S AND 0'S
907 TEST 6: CELLS' VOLATILITY TEST
908 TEST 7: SHIFTING DIAGONAL
909 TEST 10: READ RECOVERY GALLOPING TEST THROUGH EVERY 64TH CELL
910 TEST 11: READ RECOVERY LONG GALLOPING/FAST GALLOPING TEST
911 TEST 12: WORST CASE TESTING FOR CORE MEMORY
912 TEST 13: WRITE RECOVERY TEST
913
914
915 [10.0] RXDP & ACT11 & APT OPERATION
916
917 RXDP CHAIN MODE
918 -----
919
920 OPERATION IS IDENTICAL TO STAND ALONE EXCEPT:
921
922 1. NO "CZKMAF" TITLE IS PRINTED.
923 2. NO TEST 13 PRINTOUTS SUCH AS "TST13 BNK 00".
924 3. THE PROGRAM ALWAYS HALTS ON ERROR.
925 4. AT THE END OF TEST (\$ENDAD) CONTROL IS RETURNED TO
926 THE RXDP CHAIN MONITOR VIA LOCATION 42.
927 5. IF 30K SYSTEM ONLY 28K WILL BE TESTED IN XXDP CHAIN MODE
928
929 ACT11
930 -----
931
932 OPERATION IS IDENTICAL TO STAND ALONE EXCEPT:
933
934 1. NO PRINTOUTS EXCEPT ERROR PRINTOUTS.
935 2. THE PROGRAM ALWAYS HALTS ON ERROR.
936 3. AT THE END OF TEST (\$ENDAD) CONTROL IS RETURNED TO
937 THE ACT11 MONITOR VIA LOCATION 42.
938
939 APT
940 ---
941
942 OPERATION IS SIMILAR TO STAND ALONE EXCEPT:
943

944 1. THE SOFTWARE SWITCH REGISTER BECOMES LOCATION 422 (\$SWREG).
945 2. AUTO SIZING CAN BE INHIBITED BY SETTING BIT 7 OF BYTE
946 LOCATION 421 (\$ENVM).
947 3. ALL PRINTOUTS CAN BE INHIBITED BY SETTING BIT 5 OF
948 BYTE LOCATION 421 (\$ENVM).
949 4. ALL ERRORS CAUSE LOCATION 400 (\$MSGTY) TO BE SET =
950 0001 AND THE PROGRAM HALTS AT LOCATION 6240 (FATHLT).
951 LOCATION 402 (\$FATAL) CONTAINS THE ERROR NO. IN THE
952 LOW BYTE AND THE FAILING MEMORY BANK NO. IN THE HIGH
953 BYTE.
954 NOTE: THE ENVIRONMENTAL MODE BYTE SHOULD BE SET TO 240 WHILE THE SOFTWARE
955 ENVIRONMENTAL BYTE SHOULD BE SET TO 001.
956
957
958
959 .ENDR

```

960 .ENABL ARS
961 .NLIST MD,MC,CND
962
963 .LIST MF,BIN,SEG,LOC
964 .TITLE CZKMA
965 ;*COPYRIGHT (C) MARCH 1979
966 ;*DIGITAL EQUIPMENT CORP.
967 ;*MAYNARD, MASS. 01754
968 ;*
969 ;*PROGRAM BY DIAGNOSTIC ENGINEERING
970 ;*
971 ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
972 ;*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
973 ;*
974 ;$SWP=160000 ;HALT ON ERROR, LOOP ON TEST, INHIBIT ERROR TYP0UT
975
976
977
978 ;;TRAP CATCHER OF .+2 AND HALT FOR 0-776 LOCATIONS
979
980
981
982
983 SCOPE =NOP
984
985 .=42
986 .WORD 0 ;FOR ACT/XXDP
987
988 .SBTTL ACT11 HOOKS
989
990 ;;*****
991 ;HOOKS PFQUIRED BY ACT11
992 ;$SVPC=. ;SAVE PC
993 .=46
994 .SENDAD ;;1)SFT LOC.46 TO ADDRESS OF SENDAD IN .SEOP
995 .=52
996 .WORD 40000 ;;2)SET LOC.52 TO 40000
997 .=$SVPC ;; RESTORE PC
998
999 .=70
1000 000070 012737 000136 000024 PWRDN: MOV #PWRUP,0#24
1001 000076 000000
1002
    
```

```

1003
1004
1005 000104 .=104
1006 ; GET HPE IF AN ILLEGAL TRAP TO LOC. 4 OCCURRED.
1007 000104 013727 000001 000400 BUSFPI: MOV #1,#MSGTY ;TELL APT FATAL ERROR#000
1008 000112 000000 HALT ;*ERROR* TRAP TO LOC. 4 OCCURRED.
1009 ;114 AND 116 ARE RESERVED FOR PARITY TRAP VECTORS. SETUP IN
1010 ;ROUTINE "BEGIN".
1011 000120 .=120
1012
1013
1014
1015 ;* WRITE MEMORY BACKGROUND
1016 ;* -----
1017 ;*
1018 ;* THIS ROUTINE IS USED TO WRITE THE MEMORY BACKGROUND TO
1019 ;* THE VALUE STORED AT LOCATION BAKPAT. THE ROUTINE ASSUMES
1020 ;* THAT P4 IS POINTING TO THE LOWEST LOCATION AND P5 TO THE
1021 ;* HIGHEST LOCATION TO BE WRITTEN. THE PROGRAM LEAVES THE
1022 ;* SUBROUTINE WITH R0 CONTAINING THE CONTENTS OF BAKPAT.
1023 ;*
1024
1025 000120 010401 WRTHEN: MOV R4,R1 ;SET R1 TO LOWEST LOCATION UNDER TEST
1026 000122 013700 000316 MOV #BAKPAT,R0 ;LOAD R0 WITH THE CONTENTS OF LOCATION BAKPAT
1027 000126 010021 28: MOV R0,(R1)+ ;STARTING FROM THE LOWEST LOCATION WRITE THE
1028 000130 020105 CMP P1,P5 ;MEMORY TO BACKGROUND PATTERN
1029 000132 103775 BLO 28
1030 000134 000207 RTS PC ;RETURN FROM THE SUBROUTINE
1031
1032
1033 000136 013706 000350 PWRUP: MOV #SAVR6,SP ;RESTORE STACK POINTER
1034 000142 012700 006112 MOV #PNTMES-BEGIN,R0
1035 000146 060000 ADD SP,R0 ;GET THE INDIRECT ADDRESS OF LOCATION TPCRLF
1036 ;RELATIVE TO LOCATION OF DIAGNOSTIC IN THE CORE
1037 000150 004710 JSP PC,(R0) ;GO TO THE TYPE ROUTINE AND TYPE CR, LF AND A "P"
1038 000152 000120 .ASCIZ /P/
1039 .EVEN
1040
1041 000154 000411 BR START
1042
1043 ;* SERVICE XXDP/ACT11
1044 000156 004710 SENDAD: JSP PC,(R0) ;RETURN TO ACT11/XXDP MONITOR
1045 000160 000240 NOP ;IF QUICK VERIFY=RESET ELSE NOP
1046 000162 000240 NOP ;IF QUICK VERIFY=CLP #=1 ELSE INC #0
1047 000164 000240 NOP ;IF QUICK VERIFY=BR #=4 ELSE NOP
1048 000166 000430 BR .RESTART ;REPEAT TEST UNDER ACT11/XXDP
1049
1050 .=176
1051 000176 000000 SWREG: .WORD 0
1052
1053
1054 ;;*****
1055 .SBTTL START AND RESTART ROUTINES
1056 ;* RESTART AT 200 TO CLEAR APT TABLES
1057 ;;*****
1058 000200 013706 000350 START: MOV #SAVR6,SP ;SETUP STACK POINTER,
    
```

```

1059 000204 012703 000412      MOV    #UNIT,R3      ;CLEAR THE APT MAILBOX FROM $MAIL TO $DEVCT
1060 000210 000043      CLR    -(R3)         ;CLEAR A MAILBOX LOCATION
1061 000212 022703 000400      CMP    #MAIL,R3     ;DONE?
1062 000216 001374      BNE    10            ;BRANCH IF NO
1063 000220 105737 000042      TSTB  #42          ;ACT11 MODE?
1064 000224 001011      BNE    RFBRT        ;BRANCH IF YES
1065 000226 105737 000405      TSTB  ##TESTN+1    ;ARE WE RELOCATED?
1066 000232 100406      BMI    RFBRT        ;RR IF YES" SINCE TPCRLF IS RELOCATED ALG0-
1067 000234 004767 006344      JSR   PC,TPCRLF    ;PRINT TITLE
1068 000240 005103 046513      .ASCIZ /CZKMAP/
1069 000246 000060
1070
1071
1072 000250 012704 007744      RESTRT: MOV    #FNDRPG,R4      ;LOAD R4 WITH THE ADDRESS OF THE END OF THE PROGRAM
1073 000254 012703 000346      MOV    $SAVR5,R3     ;CAUSE R3 TO POINT TO THE LOCATION $SAVR5
1074 000260 012305      MOV    (R3)+,R5      ;RESTORE R5
1075 000262 012306      MOV    (R3)+,SP      ;AND RESTORE R6 JUST IN CASE IT IS A RESTART
1076 000264 010600      MOV    SP,R0         ;PLACE THE STARTING ADDRESS OF THE TEST IN R0
1077 000266 012746 000340      MOV    #340,-(SP)   ;SET HIGH PRIORITY FOR RTI
1078 000272 010046      MOV    R0,-(SP)
1079 000274 000002      RTI
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
    ;*****
    ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
    ;*****
    .SX=      ;SAVE CURRENT LOCATION
    .24      ;SET POWER FAIL TO POINT TO START OF PROGRAM
    .200     ;FOR APT START UP
    .44      ;POINT TO APT INDIRECT ADDRESS PNTR.
    $APTHDR  ;POINT TO APT HEADER BLOCK
    .=.SX    ;RESET LOCATION COUNTER
    ;*****
    ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
    ;INTERFACE SPEC.
    $APTHD:  .WORD 0
    $HIBTS:  .WORD 0 ;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
    $MBADR:  .WORD #MAIL ;ADDRESS OF APT MAILBOX (BITS 0-15)
    $TSTMI:  .WORD 000. ;RUN TIME OF LONGEST TEST
    $PASTMI: .WORD 1200. ;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
    $UNIM:   .WORD #TEND-$MAIL/2 ;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
    .WORD #ETEND-$MAIL/2 ;LENGTH MAILBOX-ETABLE(WORDS)

    REL=$TESTN+1 ;IT WILL BE 0 IF THE PROGRAM IS IN THE LOWER
                  ;CORE. BIT 7 OF THE BYTE WILL BE SET IF THE

```

```

1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
    ;PROGRAM IS IN A RELOCATED STATE AND BIT 5
    ;WILL BE SET IF PARITY BITS ARE BEING TESTED
    $MVAVAL .=$APTHD ;THIS BYTE IS USED TO DETERMINE IF MEMORY
                    ;MANAGEMENT IS AVAILABLE OR NOT
    $TYPENB .=$MVAVAL+1 ;THIS BYTE IS USED TO DETERMINE IF THE
                    ;TYPE OUT OF ERROR HAS BEEN ENABLED OR NOT
    $SPRERR .=$TYPENB+1 ;THIS BYTE DETERMINES IF THE PROGRAM HAS FOUND
                    ;A PARITY ERROR
    $ADERR .=$SPRERR+1 ;THIS BYTE IS USED TO DETERMINE IF THE
                    ;PROGRAM HAS ENCOUNTERED ADDRESS ERROR
    $STPTDI .=$ADERR+1
    $LOWBNK .=$STPTDI+2
    $PASFLG .=$LOWBNK+2 ;LOWER BYTE OF THIS WORD GIVES THE PASS NUMBER FOR
                    ;THE SPECIFIC TEST WHEREAS THE UPPER BYTE
                    ;HAS BEEN USED BY DIFFERENT TEST FOR DIFFERENT PURPOSES
    $ENDSTK .=$PASFLG+2
    $PNBK .=$ENDSTK+2 ;HOLDS BANK UNDER TEST FOR "TST BNR XX" PRINTOUT.
    $TYPCNT .=$PNBK+2 ;THIS BYTE DETERMINES THE NUMBER OF WORDS
                    ;TO BE TYPED
    $SAVKBB .=$TYPCNT+2 ;THIS LOCATION IS USED TO SAVE THE CHARACTER
                    ;HIT BY THE OPERATOR
    .EVEN
    $TKS= 177560
    $KBB= 177562
    $TPS= 177564
    $TPB= 177566
    $R0P= 177572
    $BKPAT: .WORD 377 ;BACKGROUND PATTERN WRITTEN TO MEMORY.
    $SWPAT: .WORD
    $RELBOT: .REGIN=$P ;HOLDS LOWEST TEST ADDRESS WHEN RELOCATED.
    ;*****
    ;LOCATIONS TO BE MODIFIED IF LIMITS SET BY OPERATOR
    $LOWNO: 0 ;HOLDS BITS 17:16 OF LOW TEST ADDRESS
    $LOWADD: 0 ;HOLDS BITS 15:0 OF LOW TEST ADDRESS
    $HIGHNO: 0 ;HOLDS BITS 17:16 OF HIGH TEST ADDRESS

```

```

1171 000332 037776 HIGHADD: 37776 ;HOLDS BITS 15:0 OF HIGH TEST ADDRESS
1172 ;*****
1173
1174 000334 000000 $HIMAX: 0 ;HOLDS BITS 17:16 OF MAXIMUM AVAILABLE MEMORY
1175 000336 017776 $MAXM: 17776 ;HOLDS BITS 15:0 OF MAXIMUM AVAILABLE MEMORY
1176
1177 000340 000000 MAXMEM: .WORD ;MAXIMUM CURRENT VIRTUAL MEMORY UNDER TEST
1178
1179 000342 000000 SAVMAX: .WORD
1180 000344 000000 SAVR4: .WORD
1181 000346 000000 SAVR5: .WORD
1182
1183 ;* SAVR6 POINTS TO WHERE THE PROGRAM STARTS EVEN WHEN RELOCATED.
1184 000350 000500 SAVR6: .WORD R6GIN ;CONTAINS START ADDRESS WHEN RELOCATED ALSO.
1185 000352 000000 PARMAP: 0 ;MAP OF PARITY MODULES UNDER TEST.
1186 000354 000000 SAVLOC: 0 ;TEST 6 STORES ERROR INFO HERE
1187 000356 000000 PARSP: 0 ;SAVE SP DURING PARITY ERROR TRAP.
1188 000360 000000 PAPPS: 0 ;SAVE PSW DURING PARITY ERROR TRAP.
1189 ;NOTE-PARSP +PARPS ARE NEEDED SINCE THERE IS
1190 ;IS NOT ENOUGH ROOM ON THE STACK (500-452) AND
1191 ;SO THE STACK MUST BE RESET IN THE PARERR ROUTINE.
1192 ;IN THIS CRUDE FASHION.
1193
1194
1195 ;*364-400 IS USED AS A STACK AREA BY ERPCHK ROUTINE FOR ERROR HISTORY PRINTOUT
    
```

```

1196 000400 .=400
1197 .SRTTL APT MAILBOX=ETABLE
1198
1199 ;*****
1200 .EVEN
1201 000400 $MAIL: ;APT MAILBOX
1202 000400 000000 $MSGTY: .WORD AMSGTY ;MESSAGE TYPE CODE
1203 000402 000000 $FATAL: .WORD AFATAL ;FATAL ERROR NUMBER
1204 000404 000000 $TESTN: .WORD ATESTN ;TEST NUMBER
1205 000406 000000 $PASS: .WORD APASS ;PASS COUNT
1206 000410 000000 $DEVCT: .WORD ADEVCT ;DEVICE COUNT
1207 000412 000000 $UNIT: .WORD ANIT ;I/O UNIT NUMBER
1208 000414 000000 $MSGAD: .WORD AMSGAD ;MESSAGE ADDRESS
1209 000416 000000 $MSGLG: .WORD AMSGLG ;MESSAGE LENGTH
1210 000420 $ETABLE: ;APT ENVIRONMENT TABLE
1211 000420 0000 $ENV: .BYTE ANV ;ENVIRONMENT BYTE
1212 000421 0000 $ENVM: .BYTE ANVM ;ENVIRONMENT MODE BITS
1213 000422 000000 $SWREG: .WORD ASWREG ;APT SWITCH REGISTER
1214 000424 000000 $USWR: .WORD ASWR ;USFR SWITCHES
1215 000426 000000 $CPUOPT: .WORD ACPUOPT ;CPU TYPE,OPTIONS
1216 ;*
1217 ;* 11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
1218 ;* 11/70=06,PDQ=07,Q=10
1219 ;*
1220 ;* BIT 10=REAL TIME CLOCK
1221 ;* BIT 9=FLOATING POINT PROCESSOR
1222 ;* BIT 8=MEMORY MANAGEMENT
1222 000430 0000 $MAMS1: .BYTE AMAMS1 ;HIGH ADDRESS,M.S. BYTE
1223 000431 0000 $MTYP1: .BYTE AMTYP1 ;MEM. TYPE,BLK#1
1224 ;*
1225 ;* MEM. TYPE BYTE -- (HIGH BYTE)
1226 ;* 900 NSEC CORE=001
1227 ;* 300 NSEC RIPOLAR=002
1228 ;* 500 NSEC MOS=003
1228 000432 000000 $MADR1: .WORD AMADR1 ;HIGH ADDRESS,BLK#1
1229 ;*
1229 ;* MEM. LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
1230 000434 0000 $MAMS2: .BYTE AMAMS2 ;HIGH ADDRESS,M.S. BYTE
1231 000435 0000 $MTYP2: .BYTE AMTYP2 ;MEM. TYPE,BLK#2
1232 000436 000000 $MADR2: .WORD AMADR2 ;MEM. LAST ADDRESS,BLK#2
1233 000440 0000 $MAMS3: .BYTE AMAMS3 ;HIGH ADDRESS,M.S. BYTE
1234 000441 0000 $MTYP3: .BYTE AMTYP3 ;MEM. TYPE,BLK#3
1235 000442 000000 $MADR3: .WORD AMADR3 ;MEM. LAST ADDRESS,BLK#3
1236 000444 0000 $MAMS4: .BYTE AMAMS4 ;HIGH ADDRESS,M.S. BYTE
1237 000445 0000 $MTYP4: .BYTE AMTYP4 ;MEM. TYPE,BLK#4
1238 000446 000000 $MADR4: .WORD AMADR4 ;MEM. LAST ADDRESS,BLK#4
1239 000450 $ETEND:
1240 .MEXIT
1241
1242 ;*****
1243 .SRTTL BEGIN OF AREA TESTED (+20) WHEN PROGRAM RELOCATES.
1244
1245
    
```


CZKMA MACY11 30A(1052) 05-MAR-79 09102 PAGE 27
 CZKMAF.P11 05-MAR-79 09102 BEGIN OF AREA TESTED (+20) WHEN PROGRAM RELOCATES. SEQ 0027

```

1356 001072 022626 48: CMP (SP)+,(SP)+ ;RESTORE THE STACK POINTER
1357 001074 004767 58: JSP PC,MMEMNG ;SERVICE MEMORY MANAGEMENT IF IT IS AVAILABLE
1358 ;AND IF IT HAS TO BE TESTED
1359 001100 105737 000276 TSTP R,MFAVA ;SEE IF MEMORY MANAGEMENT HAS TO BE TESTED
1360 001104 001416 RFQ 128 ;IF NO MEM. MANG. THEN GO TO 128
1361 001106 012710 68: MOV #08,(R0) ;SET UP THE RETURN ADDRESS FROM TRAP TO 88
1362 001112 012701 020000 MOV #20000,R1 ;BEGIN CHECKING MEMORY ABOVE 28K
1363 001116 000745 BR 28
1364 001120 022626 88: CMP (SP)+,(SP)+ ;RESTORE STACK POINTER
1365 001122 022701 160000 CMP #160000,R1 ;IF R1 DID NOT READ ALL THE LOCATIONS POINTED BY
1366 ;PAGE ADDRESS REGISTER 6 THEN IT HAS REACHED THE
1367 ;MAXIMUM AVAILABLE MEMORY
1368 001126 001005 RNE 128 ;IN WHICH CASE GO TO 128
1369 001130 013702 172352 MOV #172352,R2 ;PREPARE TO UPDATE MEMORY MANAGEMENT REGISTERS
1370 001134 004767 005730 JSP PC,MMREC ;OTHERWISE GO TO UPDATE MEM. MANG. REGISTERS
1371 001140 000762 BR 68
1372 001142 024341 128: CMP -(R3),-(R1) ;CAUSE R3 TO POINT TO LOCATION $MAXM AND R1
1373 ;TO THE MAXIMUM AVAILABLE MEMORY
1374 001144 004767 000072 JSF PC,PUTADP ;GO TO THE SUBROUTINE TO PLACF THE ADDRESS IN R1
1375 ;AT LOCATIONS $MAXM AND $HIMAX
1376 001150 024343 000000 CMP -(R3),-(R3) ;MAKE R3 POINT TO HIGHADD
1377 001152 004767 000064 JSP PC,PUTADP ;PLACF THE ADDRESS IN R1 AT LOCATIONS HIGHADD
1378 ;AND HIGHTWO
1379 001156 005743 TST -(R3)
1380 001160 005043 CLR -(R3) ;CLEAR THE LOCATION LOWADD
1381 001162 005043 CLR -(R3) ;AND LOWTWO
1382 001164 012701 000104 TYPESIZ: MOV #08EP,(R0)+ ;SET UP VECTOP FOR ANY FUTURE TRAP
1383 001170 010403 MOV R1,R3 ;SET R3 TO POINT TO THE LOWEST AVAILABLE MEMORY
1384 ;LOCATION
1385 001172 012701 000324 MOV #LOWTWO,P1
1386 001176 004767 005370 JSP PC,PCRLF ;TYPE CR/LF
1387 001202 004767 005536 JSP PC,OCTTYP ;TYPE LOW TEST ADDRESSE (LOWTWO+LOWADD)
1388 001206 004767 005272 TYPHEMI: JSR PC,STYPER ;TYPE "-"
1389 001212 000055 ,ASCIZ /-/
1390 ;EVEN
1391 001214 004767 005524 JSP PC,OCTTYP ;TYPE HIGHEST TEST ADDRESS (HIGHTWO+HIGHADD)
1392 001220 012703 000330 SETSTK: JSR PC,HIGHTWO,R3 ;MAKE R3 POINT TO THE HIGH ORDER BITS OF TOP ADDRESS
1393 001224 004767 000116 MOV PC,SGTSIZ ;GET THE BITS 13-17 OF THE TOP ADDRESS
1394 ;PLACED IN BITS 0-4 OF R2
1395 001230 010401 MOV R4,P1 ;SET R1 TO LOWEST TEST ADDRESS
1396
1397 001232 062704 000022 48: ADD #18,,P4 ;APPEND THE ERROR STACK FOR THE MEMORY UNDER
1398 ;TEST TO THE END OF THE PROGRAM
1399
1400 001236 005302 DEC R2
1401 001240 002374 HGE 48
1402 001242 022737 167776 000336 CMP #167776,0*0*MAXM ;CHECK IF THIS IS A 30K SYSTEM
1403 001250 001002 RNE 58 ;BRANCH IF NOT
1404 001252 062704 000022 ADD #18,,P4 ;SAVE ANOTHER BANKS WORTH OF ERROR STACK
1405 001256 010437 000310 MOV P1,0*FFNDSTK ;SAVE THE ADDRESS OF THE END OF THE ERROR STACK
1406 001262 005021 CLR (R1)+ ;CLEAR THE ERROR STACK
1407 001264 020104 CMP R1,P4
1408 001266 001775 RLOS 68
1409 001270 012737 157776 000340 MOV #157776,0*MAXMEM ;SET MAXMEM TO MAXIMUM VIRTUAL ADDRESS
1410 001276 005723 TST (R3)+ ;TESTING MEMORY MANAGEMENT?
1411 001300 001005 RNE 58 ;BRANCH IF YES (GO SAVE LOADERS AT TOP OF VIRTUAL MEMORY)
1412 001302 021327 170000 CMP #170000 ;IS THE VIRTUAL ADDRESS ABOVE 167776?

```

CZKMA MACY11 30A(1052) 05-MAR-79 09102 PAGE 28
 CZKMAF.P11 05-MAR-79 09102 BEGIN OF AREA TESTED (+20) WHEN PROGRAM RELOCATES. SEQ 0028

```

1412 001306 103002 BHIS 68 ;BRANCH IF YES (GO SAVE LOADERS)
1413 001310 011363 000000 MOV (R3),2(P3) ;OTHERWISE MAKE THE CONTENTS OF LOCATION *MAXMEM
1414 ;EQUAL TO THE MAXIMUM AVAILABLE MEMORY
1415 ;AND FALL INTO SAVE LOADERS.
1416
1417 001314 004767 000100 SAVLDPI: JSP PC,CLPMM ;DISABLE THE MEMORY MANAGEMENT UNIT
1418 001320 005723 TST (R3)+ ;MAKE R3 TO POINT TO THE LOCATION MAXMEM
1419 001322 011305 MOV (R3),P5 ;P5 CONTAINS THE ADDRESS OF MAXIMUM AVAILABLE MEM.
1420
1421 ;IF ONLY 4K BEING TESTED DON'T SAVE LOADERS
1422
1423 001324 020527 017776 CMP R5,017776 ;ONLY TESTING 4K MAX?
1424 001330 103416 RLO 48 ;BRANCH IF YES (DON'T SAVE LOADERS)
1425
1426 001332 162705 000276 38: SUB #276,R5 ;PREPARE TO SAVE 300 BYTES OF THE LOADERS
1427 001336 005737 000042 TST R#42 ;IF RUNNING UNDER XXDP
1428 001342 001406 RFQ 28 ; THEN CONTINUE
1429 001344 023737 000042 000046 CMP #28,0*46 ; ELSE IF NOT UNDER XXDP CHAIN MODE
1430 001352 001406 RFQ 28 ; THEN CONTINUE
1431 001354 162705 005376 SUB #1502,*2*-276,R5 ; ELSE SAVE 1500. WORDS FOR XXDP CHAIN MODE
1432 001360 012524 28: MOV (R5)+,(R4)+ ;SAVE LOADER
1433 001362 020513 CMP P5,(R3)
1434 001364 001775 RLOS 28
1435 001366 012323 48: MOV (R3)+,(R3)+ ;SAVE THE CONTENTS OF LOCATION MAXMEM IN SAVMAX
1436 001370 010423 MOV P4,(R3)+ ;AND THE CONTENTS OF R4 AT SAVR4
1437
1438 001372 010537 000346 TSTRELI: MOV R5,0*SAVR5 ;SAVE HIGHEST VIRTUAL ADDRESS+2
1439 001376 004767 000016 TSTSIZ: JSP PC,CLPMM ;GO TO DISABLE MEMORY MANAGEMENT UNIT
1440 001402 005745 TST -(R5) ;SET P5 BACK TO HIGHEST VIRTUAL ADDRESS
1441 001404 012703 000324 18: MOV #LOWTWO,P3 ;PREPARE TO LOAD R4 AND R5 WITH THE MEMORY BOUNDRIES
1442 001410 005723 TST (R3)+ ;IF THE BITS 16,17 OF THE LOWEST LOCATION UNDER
1443 ;TEST ARE NON ZERO
1444 001412 001003 RNE 28 ;THEN GO TO 28
1445 001414 021327 157776 CMP (R3),#157776 ;IF THE LOWEST LOCATION UNDER TEST IS HIGHER THAN
1446 ;157776 THEN GO TO TEST MEMORY MANAGEMENT
1447
1448 001420 103411 RLO 48
1449 001422 032777 010000 177020 28: BIT #10000,0*SWR ;IS MEMORY MANAGEMENT SELECTED?
1450 001430 001003 RNE 38 ;YES ALL IS WELL
1451 001432 004767 000450 JSP PC,FATERR ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
1452 001436 000003 3 ;*****ERROR NUMBER 3*****
1453
1454 001440 000167 003512 38: JMP TSTMM ;GO TO TEST MEMORY MANAGEMENT
1455 001444 020423 48: CMP R4,(R3)+ ;COMPARE TOP OF PROGRAM (WITH SAVED LOADERS) TO
1456 ;LOWEST LOCATION UNDER TEST
1457
1458 001446 103002 BHIS 68
1459 001450 016304 177776 MOV #2(P3),P4 ;ADJUST R4 TO POINT TO THE LOWEST LOCATION UNDER TEST
1460 001454 005723 68: TST (R3)+ ;IF BITS 16-17 OF HIGHEST LOCATION TO BE TESTED
1461 001456 001003 RNE 88 ;ARE NON ZERO THEN GO TO 88
1462 001460 021305 CMP (R3),R5 ;OTHERWISE SEE IF THE HIGHEST LOCATION TO BE
1463 ;TESTED IS HIGHER THAN HIGHEST VIRTUAL ADDRESS
1464 001462 010001 BHI R5 ;IF SO THEN GO TO 88
1465 001464 011305 MOV (R3),R5 ;MODIFY R5
1466 001466 105737 000405 88: TSTR #REL ;ARE WE RELOCATED.?
1467 001472 100014 RPL 108 ;BRANCH IF NO
1468 001474 013704 000322 MOV 0*RELBOT,R4 ;SET BOTTOM TEST ADDRESS WHEN RELOCATED.

```

```

1468 001500 020527 017776      CMP    R5,#17776      ;ARE WE RELOCATED IN BANK 0?
1469 001504 103402              BLO   9#              ;BRANCH IF YES
1470 001506 012705 017776      MOV    #17776,R5      ;ELSE SET HIGH MEMORY UNDER TEST=4K
1471
1472 001512 020405              9#1   CMP    R4,R5      ;IS LOW LIMIT LOWER THAN HIGH LIMIT?
1473 001514 103403              BLO   10#             ;BRANCH IF YES
1474 001516 004767 004444      JSR   PC,FATEPR      ;REPOR+ REPORT ERROR MESSAGE AND HALT AT FATHLT
1475 001522 000004              4      ;*****ERROR NUMBER 4*****
1476
1477 001524 012703 000342      10#1  MOV    #SAVMAX,R3     ;RESTORE THE CONTENTS OF MAXMEM
1478 001530 011343              MOV    (R3),*(R3)     ;MAKE THE CONTENTS OF MAXMEM = MAXIMUM AVAILABLE
1479 001532 002713 000002      MEMTST: ADD   #?,(R3)   ;MEMORY +2
1480
1481 001536 005725              TST   (R5)+           ;AND SET R5=MAX MEMORY+2
1482
1483 ;CLEAR MEMORY UNDER TEST
1484
1485 001540 010500      CLRMEM: MOV    R5,R0      ;MOVE HIGH ADDRESS TO R0
1486 001542 005040      2#1   CLP   -(R0)          ;REGIM CLEARING THE MEMORY FROM THE TOP
1487 001544 020004      CMP    R0,R4          ;UNTIL THE BOTTOM IS REACHED
1488 001546 101375      BHI   2#              ;
1489 001550 012702 000001      MOV    #1,R2          ;SET R2 TO ENABLE PARITY MODULE CODE,
1490 001554 004767 005740      JSR   PC,PARITY      ;ENABLE PARITY IF WANTED AND AVAILABLE.
1491 001560 012702 000316      MOV    #BAKPAT,R2     ;
1492 001564 012212      MOV    (R2)+,(R2)     ;WRITE SWAPPED BAKPAT IN LOCATION SWAPAT
1493 001566 000312      SWAB  (R2)           ;
1494 001570 017702 176654      MOV    #SWP,R2        ;LOAD R2 WITH THE OPTIONS STORED AT #SWREG
1495 001574 042702 177760      RIC   #177760,R2     ;ONLY LEAVE THE LOWER 4 BITS OF #SWREG IN R2 TO GO TO
1496 ;THE TEST # SPECIFIED [DEFAULT IS TEST#0]
1497
1498
1499
1500 ;ENTER HERE FROM TSTSCP ROUTINE AT END OF SUBTEST
1501
1502 001600 005037 000306      CONT: CLR    #PASFLG   ;INIT SUBTEST PASS FLAG,
1503 001604 110237 000404      MOVB  R2,#TESTN      ;GET UP #TEST# WITH THE TEST NUMBER GOING
1504 ;TO BE EXECUTED
1505 001610 010401      100P: MOV    R4,R1        ;LOAD R1 WITH THE LOWEST LOCATION UNDER TEST
1506 001612 010400      MOV    R4,R0          ;PLACF THE ADDRESS OF THE LOWEST LOCATION UNDER
1507 ;TEST IN R0
1508 001614 010403      MOV    R4,R3          ;AND IN R3
1509 001616 006302      ASL   R2              ;
1510 001620 000702      ADD   PC,R2           ;
1511 001622 006207 000004      ADD   TFL=,(R2),PC   ;GO TO THE TEST #
1512 ;STORED IN BITS 0-3 OF SWITCH REGISTER
1513
1514
1515 001626 000102      TBL:  TST0-TBL        ;RELATIVE ADDRESS OF TEST # 0
1516 001630 000340      TST1-TBL        ;RELATIVE ADDRESS OF TEST # 1
1517 001632 000440      TST2-TBL        ;RELATIVE ADDRESS OF TEST # 2
1518 001634 000550      TST3-TBL        ;RELATIVE ADDRESS OF TEST # 3
1519 001636 000816      TST4-TBL        ;RELATIVE ADDRESS OF TEST # 4
1520 001640 001126      TST5-TBL        ;RELATIVE ADDRESS OF TEST # 5
1521 001642 001274      TST6-TBL        ;RELATIVE ADDRESS OF TEST # 6
1522 001644 001430      TST7-TBL        ;RELATIVE ADDRESS OF TEST # 7
1523 001646 001654      TST10-TBL       ;RELATIVE ADDRESS OF TEST # 10
  
```

```

1524 001650 002204      TST11-TBL       ;RELATIVE ADDRESS OF TEST # 11
1525 001652 002256      TST12-TBL       ;RELATIVE ADDRESS OF TEST # 12
1526 001654 002530      TST13-TBL       ;RELATIVE ADDRESS OF TEST # 13
1527 001656 003156      RFLOC-TBL       ;RELATIVE ADDRESS OF ROUTINE "RELOC"
1528
1529
1530
1531 ;R5 IS POINTING TO THE TOP OF THE MEMORY TO BE TESTED+2
1532 ;R4 & R0 ARE POINTING TO THE LOWEST ADDRESS OF MEMORY TO BE TESTED
  
```



```

1533 ;* SCOPE ROUTINE
1534 ;* -----
1535 ;*
1536 ;*
1537 ;*
1538 ;* PROGRAM COMES TO THIS ROUTINE AFTER COMPLETION OF EACH TEST AND
1539 ;* IF CNTRL-C TYPED GOTO ERROR HISTORY TYPE ROUTINE.
1540 ;* IF SR= 2000 (BIT10) THEN HALT
1541 ;* IF SR= 40000 (BIT14) THEN LOOP ON TEST DEFINED BY SP BITS<310>
1542 ;* ELSE CONTINUE TO NEXT TEST.
1543 ;*
1544 ;*
1545 001660 105737 000420 TSTSCP: TSTB #6ENV ;ARE WE RUNNING UNDER APT?
1546 001664 001002 BNE CNTSCP ;IF SO THEN GO TO CNTSCP
1547 001666 004767 JSR PC,CHECKC ;TEST FOR CONTROL-C AND IF TYPED GO
1548 ;PRINT ERROR HISTORY AND HALT AT FAHLT.
1549 001672 113702 CNTSCP: MOVB #*TESTN,R2 ;PLACE THE TEST NUMBER IN THE LOWER BYTE OF R2
1550 ;SINCE THERE ARE LESS THAN 377 TESTS UPPER BYTE
1551 ;OF R2 WILL BE 0
1552 001676 005237 000410 INC #*DEVCT ;TELL APT WE ARE STILL RUNNING OKAY
1553 001702 032777 002000 176540 BIT #2000,*5WR ;IS THE PROGRAM GOING TO HALT AFTER EACH TEST?
1554 001710 001401 BEQ TSTGO ;IF NOT THEN GO TO 28
1555 001712 000000 SWHALT: HALT ;HALT AT END OF TEST SWITCH SET.
1556 ;*
1557 001714 032777 040000 176526 TSTGO: BIT #40000,*5WR ;IS THE PROGRAM GOING TO LOOP ON TEST
1558 001722 001332 BNE LOOP ;IF SO THEN GO TO THE STARTING OF THE SAME TEST
1559 001724 105202 INCB P2
1560 001726 000724 RR CONT ;GO TO CONT AND CONTINUE EXECUTING THE NEXT TEST
    
```

```

1561 ;*****
1562 ;*TEST #1 TEST FOR PROPER BANK SELECTION
1563 ;*(1) THIS TEST ASSUMES THAT THE MEMORY IS IN A STATE
1564 ;* OF ALL 0'S AND R0 HAS THE ADDRESS OF THE LOWEST
1565 ;* LOCATION UNDER TEST
1566 ;*(2) IT CHECKS FOR PROPER BANK SELECTION BY WRITING
1567 ;* 1'S IN A LOCATION AND CHECKING FOR 0'S IN THE SAME
1568 ;* LOCATIONS OF OTHER 4K BANKS OF THE MEMORY
1569 ;* (I.E. LOCATIONS LIKE 7766 AND 27766 ETC.)
1570 ;*(3) THIS TEST ALSO CHECKS TO SEE THAT NONE OF THE NON EXIST-
1571 ;* ING BANK RESPOND WHEN THEY ARE ADDRESSED
1572 ;*****
1573 001730 105737 000404 TST#1: TSTR #*TESTN ;CHECK FOR PROPER TEST SEQUENCE
1574 001734 001403 BEQ .+10
1575 001736 004767 JSR PC,SEOFFR ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FAHLT
1576 001742 000005 S ;*****ERROR NUMBER 5*****
1577 ;*
1578 001744 012703 177777 MOV #177777,R3
1579 001750 010401 MOV R4,P1 ;R1 = ADDRESS OF LOWEST LOCATION OF MEMORY UNDER TEST
1580 001752 010310 MOV R3,(R0) ;SET ALL THE BITS AT (R0)
1581 001754 020001 CMP R0,P1 ;IS R0 POINTING TO THE SAME MEMORY LOCATION AS R1
1582 001756 000417 BEQ 48 ;IN WHICH CASE CHECK FOR ALL 1'S AT (P1)
1583 001760 005711 TST (R1) ;OTHERWISE CHECK (R1) FOR ALL 0'S
1584 001762 001430 BFO 58
1585 001764 020311 CMP R3,(R1) ;IF R1 IS NOT EQUAL TO R0 AND (R1)
1586 ;DOES NOT CONTAIN ALL 0'S THEN
1587 ;CHECK TO SEE IF (R0) = (P1)
1588 001766 001004 BNE 38
1589 001770 012767 MOV #6,126 ;*ERROR* SETUP ERROR NO. IN 126
1590 ;*****ERROR NUMBER 6*****
1591 001776 000403 BR 106
1592 002000 38:
1593 002000 012767 000007 000032 MOV #7,126 ;*ERROR* SETUP ERROR NO. IN 126
1594 ;*****ERROR NUMBER 7*****
1595 002006 010046 100: MOV R0,*(SP) ;SAVE R0 ON STACK
1596 002010 105237 000301 INCB #*ADDRERR ;AN ADDRESSING ERROR IS SUSPECTED
1597 002014 000407 BP 116
1598 002016 020311 48: CMP R3,(R1) ;CHECK (P1) FOR ALL 1'S
1599 002020 001411 BEQ 58
1600 002022 012767 000010 000010 MOV #10,126 ;*ERROR* SETUP ERROR NO. IN 126
1601 ;*****ERROR NUMBER 10*****
1602 002030 010046 100: MOV R0,*(SP) ;SAVE R0 ON STACK
1603 002032 010300 MOV R3,R0
1604 002034 004767 003570 110: JSR PC,ERR0P ;GO TO THE ERROR SUBROUTINE
1605 002040 000000 120: .WORD ;ERROR NUMBER TO BE REPORTED WILL BE PLACED HERE
1606 002042 012600 MOV (SP)+,R0 ;RESTORE R0
1607 ;*
1608 002044 013706 000350 58: MOV #5SAVR6,SP ;RESTORE THE STACK POINTER
1609 002050 062701 022000 ADD #20000,R1 ;CAUSE R1 TO POINT TO THE SAME CHIP
1610 ;LOCATION IN THE NEXT 4K BANK OF MEMORY
1611 ;BY ADDING 1 TO THE 14TH BIT OF ADDRESS IN R1
1612 002054 020105 CMP R1,R5 ;COMPARE R1 WITH THE HIGHEST MEMORY
1613 ;LOCATION WHICH IS STORED IN R5
1614 002056 103736 BLO 28 ;IF R1 LESS THAN R5 THEN REPEAT THE TEST FROM 28
1615 ;*
1616 002060 105737 000421 TSTB #6ENV ;HAS APT INHIBITED SIZING?
    
```

```

1617 002064 100432      BMI      00      ;BRANCH IF YES (DON'T TEST NON-EXISTENT MEMORY)
1618 002066 032777 000100 176354    BIT      #100,0BWR ;HAS USER INHIBITED SIZING?
1619 002074 001026      RNE      00      ;BRANCH IF YES (DON'T TEST NON-EXISTENT MEMORY)
1620
1621 002076 020127 157776    CMP      R1,#157776 ;SEE IF R1 HAS CROSSED 20K BOUNDARY OF VIRTUAL ADDRESS
1622 002102 101016      BHI      00      ;IN WHICH CASE GO TO 66
1623
1624 002104 020137 000340    CMP      R1,#MAXMEM ;SHOULD BE LEFT AS IS FOR 30K SYSTEMS (WHICH USE 16K CHI
1625
1626 002110 103755      BLO      56      ;IS R1 LOWER THAN THE MAXIMUM AVAILABLE
1627 002112 012702 000006    MOV      #4,R2      ;MEMORY ?
1628 002116 012712 000340    MOV      #340,(R2)  ;IF 00 THEN GO TO 56
1629 002122 012742 177714    MOV      #56,-(R2) ;MAKE R2 POINT TO TRAP VECTOR+2 FOR NXM
1630 002126 060712      ADD      PC,(R2)    ;SET PSW TO 340
1631 002130 011111      MOV      (R1),(R1) ;SET UP RETURN ADDRESS FROM TRAP TO 56
1632 002132 004767 004030    JSR      PC,FATERR ;TRY TO WRITE TO NON-EXISTENT MEMORY (SHOULD TRAP)
1633 002136 000011      I1       ;**ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
1634
1635
1636 002140 012702 000004    66:    MOV      #4,R2
1637 002144 012722 000006    66:    MOV      #6,(R2)+ ;RESTORE TRAP VECTOR
1638 002150 005012      CLR      (R2)
1639 002152 005010      CLR      (R0)
1640
1641 002154 062700 020000    ADD      #20000,PC ;CAUSE R0 TO POINT TO THE SAME CHIP
1642
1643
1644 002160 020005      CMP      R0,R5     ;LOCATION IN THE NEXT 4K MEMORY BANK
1645
1646
1647 002162 103672      BLO      16      ;BY ADDING 1 TO THE 14TH BIT OF ADDRESS IN R0
1648 002164 000035      END0:    BR      TRTSCP    ;COMPARE R0 WITH THE HIGHEST MEMORY
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
    
```

```

1651
1652
1653
1654
1655 002166 022737 000001 002404    TST1:  CMPR      #1,#TESTN ;*****
1656
1657
1658 002174 001403      REQ      #10
1659 002176 004767 000376    JSR      PC,SEQERR ;**ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
1660 002202 000012      I2       ;*****ERROR NUMBER 12*****
1661
1662 002204 012700 000001    10:    MOV      #1,R0
1663 002210 010002      MOV      R0,R2     ;SET R2#1
1664 002212 010011      MOV      R0,(R1)  ;MOV 1 AT LOCATION (R1)
1665 002214 020011      CMP      R0,(R1)  ;COMPARE R1 WITH THE CONTENTS OF LOCATION (R1)
1666 002216 001403      BEQ      48
1667 002220 004767 001404    JSR      PC,ERR0R ;**ERROR* REPORT ERROR MESSAGE
1668 002224 000011      I3       ;*****ERROR NUMBER 13*****
1669
1670
1671 002226 000702      TST      R0       ;ARE WE SHIFTING A 0 IN DATA DIRECTION?
1672 002230 001400      BLO      58
1673 002232 000300      ASL      R1       ;IF 00 THEN GO TO 58
1674
1675 002234 103366      BCC      28      ;SHIFT THE 1 BROUGHT IN AT 18 IN
1676
1677 002236 000002      CLR      R0       ;DATA DIRECTION
1678 002240 012700 177776    MOV      #177776,R0 ;IF THE 1 HAS NOT BEEN SHIFTED THRU
1679 002244 000762      RR      28      ;INITIATE SHIFTING OF 0 IN DATA DIRECTION
1680
1681 002246 000201      SFC      R0
1682 002250 000100      ROL      R0       ;SHIFT A 0 16 TIMES IN DATA DIRECTION
1683 002252 103747      RSC      28      ;IF THE 0 HAS NOT BEEN SHIFTED THRU
1684
1685 002254 062701 020000    ADD      #20000,R1 ;IF 16 DATA BITS THEN REPEAT FROM 28
1686
1687 002260 020105      CMP      R1,R5     ;OTHERWISE GO TO THE NEXT BANK OF
1688 002262 103750      BLO      16      ;4K MEMORY AND REPEAT THE TEST
1689 002264 000737      END1:    BR      ENDP
1690
    
```

```

1691 ;*****
1692 ;*TEST 2 TEST MEMORY FOR HOLDING DATA AND BYTE SELECTION
1693 ;*(1) THIS TEST CHECKS THE MEMORY FOR THE CAPABILITY
1694 ;* OF HOLDING 1'S AND 0'S BY WRITING A BACKGROUND
1695 ;* OF BAKPAT AND READING IT
1696 ;*(2) MEMORY IS WRITTEN USING A BYTE AT A TIME
1697 ;*(3) STEPS 1 & 2 ARE REPEATED WITH A SWAPPED BACKGROUND PATTERN
1698 ;*****
1699 002266 122737 000002 000404 T5T21 CMPB #2,#*TESTN ;CHECK FOR PROPER TEST SEQUENCE
1700
1701 002274 001403 BEQ #+10
1702 002276 004767 003664 JSR PC,SEOFFR ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
1703 002302 000014 14 ;*****ERROR NUMBER 14*****
1704
1705 002304 013700 000316 15: MOV #*BAKPAT,R0
1706 002310 110021 MOVB R0,(R1)+
1707 002312 113721 000317 MOVB #*BAKPAT+1,(R1)+;WRITE THE MEMORY WITH THE WORD STORED IN BAKPAT
1708 002316 020105 CMP R1,R5
1709 002320 103771 BLO 15
1710
1711 002322 020041 25: CMP R0,-(R1) ;TEST THE MEMORY TO SEE IF IT CONTAINS
1712 ; THE WORD STORED IN BAKPAT
1713 002324 001416 BEQ R5
1714 002326 062701 000002 ADD #2,R1
1715 002332 123741 000317 CMPB #*BAKPAT+1,-(R1);CHECK FOR BYTE SELECTION PROBLEM
1716 002336 001402 BEQ 48
1717 002340 120041 CNDB R0,-(R1) ;AGAIN CHECK FOR BYTE SELECTION PROBLEM
1718 002342 001002 SNF 68
1719 002344 105237 000301 46: INCR #*ADERR ;PREPARE TO INFORM THAT IT IS ADDRESSING ERROR
1720 002350 042701 000001 68: R1C #1,R1 ;MAKE THE ADDRESS IN R1 EVEN
1721 002354 004767 003250 JSR PC,ERROR ;*ERROR* REPORT ERROR MESSAGE
1722 002356 000015 15 ;*****ERROR NUMBER 15*****
1723
1724 002362 020104 88: CMP R1,R4 ;KEEP ON TESTING THE MEMORY UNTIL
1725 002364 001356 BHI 25 ;R1 EQUALS THE LOWEST ADDRESS
1726 002366 000337 000316 SWAB #*BAKPAT ;CHANGE THE DATA PATTERN
1727 002372 001744 BEQ 15 ;IF THE DATA PATTERN DOES NOT HAVE LOW
1728 ; BYTE #0 THEN FALL THRU
1729 002374 000733 END2: RR END1
1730
1731 ;THE TEST LEAVES BAKPAT LOCATION THE SAME AS IT WAS IN THE BEGINNING
1732

```

```

1733 ;*****
1734 ;*TEST 3 DUAL ADDRESS TEST A
1735
1736 ;*(1) THIS TEST CHECKS FOR DUAL ADDRESSING PROBLEMS BY WRITING A
1737 ;* BACKGROUND OF BAKPAT.
1738 ;*(2) STARTING FROM THE LOWEST LOCATION IN THE BANK THE TEST WRITES A
1739 ;* LOCATION WITH SWAPPED BAKPAT
1740 ;*(3) READS THE MEMORY FOR PROPER CONTENTS
1741 ;*(4) SHIFTS A 1 ALONG THE ADDRESS DIRECTION AND REPEATS STEPS 1-3
1742 ;*(5) REPEATS STEP 1-4 FOR EACH 4K BANK
1743 ;*****
1744 002376 122737 000003 000404 T5T3:1 CMPB #3,#*TESTN ;CHECK FOR PROPER TEST SEQUENCE
1745 002404 001403 BEQ #+10
1746 002406 004767 003554 JSR PC,SEOFFR ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
1747 002412 000016 16 ;*****ERROR NUMBER 16*****
1748
1749 002414 005003 78: CLP R3
1750 002416 004737 000120 JSR PC,#*RTMEM ;WRITE MEMORY WITH THE BACKGROUND STORED
1751 ; AT LOCATION BAKPAT
1752 002422 005002 46: CLP R2
1753 002424 050302 68: RIS R3,R2 ;MAKE R2 POINT TO THE MEMORY BANK POINTED BY R3
1754 002426 020204 CMP R2,R4 ;IF R2 IS LESS THAN R4
1755 002430 103465 BLO 168 ;THEN DO NOTHING
1756 002432 020205 CMP R2,R5 ;IF R2 IS HIGHER THAN THE HIGHEST LOCATION TO BE
1757 002434 103077 BHS 208 ;TESTED THEN EXIT THE TEST
1758 002436 000312 SWAB (#2) ;OTHERWISE WRITE THE COMPLEMENT OF BAKPAT IN
1759 ; THE LOCATION POINTED BY R2
1760
1761 002440 005001 75: CLR R1
1762 002442 050301 RIS R3,R1
1763 002444 020104 CMP R1,R4 ;IF R1 IS POINTING TO A LOCATION LOWER THAN R4
1764 002446 103445 BLO 128 ;THEN GO TO 128
1765 002450 020105 CMP R1,R5
1766 002452 103053 BHS 158 ;CHECK THE MEMORY FOR CORRECT DATA
1767 002454 020102 CMP R1,R2
1768 002456 001431 BEQ 108
1769 002458 020011 CMP R0,(R1) ;IF R1 IS NOT = TO R2 THEN (R1) SHOULD HAVE
1770 ; THE SAME WORD AS BAKPAT
1771 002462 001437 BEQ 128 ;IN WHICH CASE GO BACK TO 128
1772 002464 012767 000017 000032 MOV #17,228 ;*ERROR* SETUP ERROR NO. IN 228
1773 ;*****ERROR NUMBER #17*****
1774 002472 010046 88: MOV R0,-(SP) ;PLACE R0 ON THE STACK
1775 002474 000316 SWAB (SP)+,(R1) ;IF (R1) IS NOT = R0 THEN SEE IF IT IS SAME
1776 002476 022611 CMP #AS A SWAPPED R0
1777 002500 001003 RNE 98 ;IF NOT THEN A SUSPECTED DUAL ADDRESSING PROBLEM
1778 ;FOR THE BITS THAT ARE DIFFERENT IN R0 AND (R1)
1779 ;OTHERWISE THERE IS DUAL ADDRESSING FOR THE
1780 ; ENTIRE WORD
1781 002502 012767 000020 000014 MOV #20,228 ;*ERROR* SETUP ERROR NO. IN 228
1782 ;*****ERROR NUMBER #20*****
1783 002510 105237 000301 98: INCR #*ADERR ;ADDRESSING PROBLEM IS DETECTED
1784 002514 010046 MOV R0,-(R0) ;SAVE R0
1785 002516 010200 MOV R2,R0 ;SET R0=GOOD ADDRESS FOR ERROR REPORT
1786 002520 004767 003104 JSR PC,ERROR ;GO TO THE ERROR SUBROUTINE
1787 002524 000000 ,WORD ;ERROR NUMBER TO BE REPORTED WILL BE PLACED HERE
1788 002526 012600 MOV (R0)+,R0 ;RESTORE R0

```

```

1789 002530 010011 MOV R0,(R1) ;RESTORE (R1)
1790 002532 020037 000316 CMP R0,#BAKPAT ;IF THE CONTROL CAME HERE FROM 150-2 THEN
1791 002536 001411 BEQ 120 ;
1792 002540 000407 RR R0 ;RETURN TO 110
1793 002544 000300 1001 SWAB R0 ;MAKE R0 SAME AS SWAPPED BAKPAT
1794 002544 020011 CMP R0,(R1) ;IF R1 = R2 THEN (R1) SHOULD CONTAIN A WORD
1795 ;EQUAL TO SWAPPED R0
1796 002546 001404 BEQ 110 ;IN WHICH CASE GO BACK TO 110
1797 002550 012767 000021 177746 MOV #21,220 ;ERROR* SETUP ERROR NO. IN 220
1798 ;*****ERROR NUMBER #21*****
1799 002556 000745 BR R0 ;AND GO TO 00
1800 002560 000300 1101 SWAB R0 ;RESTORE R0 TO BAKPAT
1801 002562 040301 1201 BIC R3,R1 ;TAKE OUT THE BANK ADDRESS FROM THE ADDRESS IN R1
1802 002564 005701 TST R1 ;IF R1 IS 0 THEN PLACE A 1 IN R1
1803 002566 001001 BNE 130 ;OTHERWISE GO TO 130
1804 002570 005201 INC R1
1805 002572 006101 1301 ROL R1
1806 002574 020127 020000 CMP R1,#20000 ;IF R1 IS LESS THAN A 4K BOUNDRY
1807 002600 103700 BLO 70 ;THEN REPEAT FROM 70
1808 002602 000312 1501 SWAB (R2) ;RESTORE (R2) TO BAKPAT
1809 002604 040302 1601 BIC R3,R2 ;TAKE OUT THE BANK ADDRESS FROM THE ADDRESS
1810 ;STORED IN R2
1811 002606 005702 TST R2 ;IF R2 = 0 THEN MOVE A 1 TO R2
1812 002610 001001 BNE 100 ;OTHERWISE GO TO 100
1813 002612 005202 INC R2
1814 002614 006102 1801 ROL R2 ;SHIFT A ONE IN THE ADDRESS WORD
1815 002616 020227 020000 CMP R2,#20000 ;IS THE ADDRESS IN R2 MORE THAN THE BOUNDRY
1816 ;OF 4K
1817 002622 103700 BLO 60 ;IF NOT THEN GO TO 60
1818 002624 060203 ADD R7,R3 ;OTHERWISE MAKE R3 POINT TO THE NEXT 4K BANK
1819 002626 020337 000340 CMP R3,#MAXMEM ;IF R3 IS POINTING TO A BANK THAT IS LOWER
1820 ;THAN MAXMEM
1821 002632 103673 BLO 40 ;THEN REPEAT FROM 40
1822 002634 000337 2001 SWAB 0#BAKPAT
1823 002640 001656 BEQ TST3 ;REPEAT THE TEST WITH SWAPPED BAKPAT ONLY IF
1824 ;THE LOWER BYTE OF BAKPAT IS 0
1825 002642 000654 END31 BR END2
    
```

```

1826 ;*****
1827 ;*TEST 4 DUAL ADDRESS TEST B
1828 ;*(1) THIS TEST CHECKS FOR DUAL ADDRESSING BY WRITING
1829 ;* AND READING THE ADDRESS IN THE LOCATION AND THEN
1830 ;* WRITING AND READING ADDRESS COMPLEMENT
1831 ;*****
1832 002644 122737 000004 000404 TST41 CMPB #4,#TST4 ;CHECK FOR PROPER TEST SEQUENCE
1833 002652 001403 REQ +10
1834 002654 004767 003306 JSP PC,SEQEPP ;ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
1835 002660 000022 ;*****ERROR NUMBER 22*****
1836
1837 002662 005003 CLP R3
1838 002664 010100 101 MOV R1,R0
1839 002666 005703 TST R3 ;IF R3 IS NOT 0 THEN STORE THE ADDRESS
1840 002670 001401 BEQ 20 ;IN THE LOCATION
1841 002672 005100 COM R0 ;OTHERWISE STORE COMPLEMENT
1842 002674 010021 201 MOV R0,(R1)+ ;OF THE ADDRESS
1843 002676 020105 CMP R1,R5 ;UNTIL THE HIGHEST MEMORY LOCATION IS REACHED
1844 002700 103771 BLO 10
1845
1846 002702 020041 301 CMP R0,(R1) ;CHECK THE LOCATION FOR THE CORRECT CONTENTS
1847 002704 001405 BEQ 40
1848 002706 105237 000301 INCB 0#ADERRP ;THIS IS PROBABLY ADDRESS PROBLEM RATHER THAN
1849 ;BIT PROBLEM
1850 002712 004767 002712 JSR PC,ERR0R ;ERROR* REPORT ERROR MESSAGE
1851 002716 000023 23 ;*****ERROR NUMBER 23*****
1852
1853 002720 010100 401 MOV R1,R0
1854 002722 162700 000002 SUR #7,R0 ;CHECK THAT THE ADDRESS IS STORED AT
1855 002726 005703 TST R3 ;LOCATION IF R3 IS NOT 0
1856 002730 001401 BEQ 50 ;OTHERWISE CHECK FOR
1857 002732 005100 COM R0 ;ADDRESS COMPLEMENT
1858 002734 020104 501 CMP R1,R4
1859 002736 101361 BHI 30
1860 002740 112737 000001 000306 MOVB #1,#PASFLG ;SET PASFLG FOR ERROR REPORT
1861 002746 005103 COM R3 ;COMPLEMENT THE CONTENTS OF R3
1862 002750 001345 BNE 10 ;REPEAT TST3 IF R3, IS NON 0, ENABLING ADDRESS
1863 ;COMPLEMENT TO BE WRITTEN AND READ, OTHERWISE FALL THRU
1864 002752 000733 END41 BR END3
1865
    
```

```

1866 ;*****
1867 ;*TEST 5 MARCHING 1'S AND 0'S
1868 ;*(1) THIS TEST WRITES A BACK GROUND OF THE WORD STORFD
1869 ;* AT BAKPAT.
1870 ;*(2) READS EVERY LOCATION FOR CORRECT DATA, SWAPS BYTES
1871 ;* AT THE LOCATION AND PROCEEDS IN MAX. TO MIN
1872 ;* DIRECTION OF MEMORY LOCATIONS.
1873 ;*(3) READS EVERY LOCATION FOR SWAPPED BAKPAT PATTERN
1874 ;* WRITES BAKPAT BACKGROUND IN THE LOCATION AND PROCEEDS
1875 ;* IN MIN. TO MAX. DIRECTION
1876 ;*(4) REPEATS STEP 2 GOING IN MIN. TO MAX. DIRECTION
1877 ;*(5) REPEATS STEP 3 GOING IN MAX. TO MIN. DIRECTION
1878
1879 ;*****
1880 TST5: CMPB #5,#8TEST5 ;CHECK FOR PROPER TEST SEQUENCE
1881
1882 002754 122737 000005 000404 BEQ #10 ;
1883 002762 001403 JSR PC,SEQRFP ;ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
1884 002770 000024 24 ;*****ERROR NUMBER 24*****
1885
1886 002772 004737 000120 18: JSR PC,#BRTMEM ;GO TO WRITE THE MEMORY WITH A BACKGROUND OF THE
1887 ;* WORD STORED IN BAKPAT
1888 002776 020041 25: CMP #0,(R1) ;READ THE CONTENTS OF LOCATION POINTED BY R1
1889 003000 001403 RFG 35 ;TO SEE IF IT HAS THE SAME VALUE AS R0
1890 003002 004767 002622 JSR PC,ERR0P ;ERROR* REPORT ERROR MESSAGE
1891 003006 000025 25 ;*****ERROR NUMBER 25*****
1892
1893 003010 000300 38: SWAP R0 ;SWAP THE BYTES AT (R1)
1894 003012 010011 MOV R0,(R1) ;READ (R1) FOR CORRECT VALUE
1895 003014 021100 CMP #0,(R1) ;
1896 003016 001403 BEQ 48 ;
1897 003020 004767 002604 JSR PC,ERR0P ;ERROR* REPORT ERROR MESSAGE
1898 003024 000026 26 ;*****ERROR NUMBER 26*****
1899
1900
1901 003026 000300 48: SWAP R0 ;SWAP THE BYTES OF THE REGISTER
1902 ;* CONTAINING BACKGROUND PATTERN
1903 003030 001023 BNE 98 ;IF THE LOWER BYTE OF THE REGISTER
1904 ;* IS NOT 0 THEN THE PROGRAM IS READING
1905 ;* THE MEMORY TO CONTAIN A BACK GROUND OF
1906 ;* BAKPAT AND WRITING THE SWAPPED WORD
1907 ;* IN WHICH CASE GO TO 98
1908
1909
1910
1911 003032 005703 58: TST R3 ;R3 WAS 0 WHEN THE PROGRAM ENTERED
1912 ;* THIS TEST, AND IT IS NOT ALTFRED UNTIL PASFLG=3
1913 ;* IF R3 EQUAL 0 THEN THE PROGRAM IS
1914 ;* READING/WRITING MIN. TO MAX. OTHERWISE
1915 ;* IT IS GOING IN MAX. TO MIN. DIRECTION
1916 003034 001023 RNE 1'S ;IF R3 IS NOT CLEAR THEN GO TO 108
1917 003036 062701 68: AND #2,R1 ;OTHERWISE ADD 2 TO THE CONTENTS OF R1
1918 003042 020105 CMP #1,R5 ;COMPARE R1 WITH THE MAX. MEMORY LOCATION TO
1919 ;* RE TESTED
1920 003044 103006 RDIS 88 ;IF R1>85 THEN GO TO 88 OTHERWISE
1921 003046 020011 CMP #0,(R1) ;READ (R1) FOR THE CORRECT DATA
    
```

```

1922 003050 001757 30 ;WRITE COMPLEMENT OF THE DATA FOUND AT (R1)
1923 ;AND REPEAT UNTIL R1 > R5
1924 003052 004767 002552 JSR PC,ERR0P ;ERROR* REPORT ERROR MESSAGE
1925 003056 000027 27 ;*****ERROR NUMBER 27*****
1926
1927 003060 000753 BR 35 ;
1928 003062 105237 000306 88: INCR #0,PASFLG
1929 003066 000300 SWAP R0 ;IF THE LOWER BYTE OF R0 IS ALL 0'S
1930 003070 001742 RFG 25 ;THEN BEGIN READING BAKPAT SWAPPED WRITING BAKPAT
1931 ;* AND READING BAKPAT GOING FROM MAX. TO MIN.(PASFLG=4)
1932 ;* OTHERWISE CLEAR R0
1933 003072 005103 COM R3 ;PUT THE LOWEST TESTING ADDRESS IN R1
1934 003074 010401 MOV #0,R1 ;AND BEGIN READING 0'S, WRITING 1'S AND
1935 003076 000763 BR 78 ;READING 1'S IN MIN. TO MAX. DIRECTION (PASFLG=3)
1936
1937
1938 003100 005703 98: TST R3 ;IF R3 IS NON 0, I.E. PASFLG=3
1939 003102 001353 RNE 58 ;THEN READ BAKPAT, WRITE
1940 ;* SWAPPED BAKPAT AND READ SWAPPED BAKPAT
1941 ;* IN MIN. TO MAX. DIRECTION
1942 003104 020104 108: CMP R1,R4 ;OTHERWISE TEST IS PROCEEDING IN MAX. TO
1943 ;* MIN. DIRECTION.
1944 003106 101333 RHT 25 ;KEEP ON LOOPING UNTIL R1=R4
1945 003110 105237 000306 88: INCR #0,PASFLG
1946 003114 000300 SWAP R0 ;IF R0 SWAPPED HAS LOWER BYTE=0
1947 003116 001753 RFG 25 ;THEN READ BAKPAT SWAPPED, WRITE BAKPAT,
1948 ;* AND READ BAKPAT GOING FROM MIN. TO MAX.
1949
1950 003120 000714 END5: BR #0
1951
    
```

```
1952 ;*****  
1953 :*TEST 6 CELLS' VOLATILITY TEST  
1954  
1955 ;*(1) THIS TEST WRITES THE MEMORY WITH A BACK GROUND OF BAKPAT  
1956 ;*(2) WITH PASFLG#0 THE TEST READS THE MEMORY FOR BAKPAT  
1957 ;* AND THEN INCREMENTS PASFLG  
1958 ;*(3) IT THEN READS/SWAPS BYTES/WITES A LOCATION X FOR  
1959 ;* OVER 2 MSEC AND THEN READS THE MEMORY FOR BAKPAT  
1960 ;*(4) REPEATS STEP 3 WITH X=X+4K UNTIL END OF MEMORY IS ENCOUNTERED  
1961 ;*(5) IT THEN INCREMENTS PASFLG AND WRITES THE MEMORY TO  
1962 ;* BAKPAT AND WITH PASFLG#2 IT READS MEMORY FOR ALL  
1963 ;* SWAPPED BAKPAT AFTER WHICH PASFLG IS INCREMENTED TO 3  
1964 ;*(6) REPEATS STEPS 3 AND 4 READING THE MEMORY FOR SWAPPED  
1965 ;* BAKPAT INSTEAD OF BAKPAT.  
1966  
1967 ;*****  
1968 003122 122737 000006 000404 TST6: CMPR #4,0#STESTN ;CHECK FOR PROPER TEST SEQUENCE  
1969  
1970  
1971 003130 001403 REQ ;+10  
1972 003132 004767 JSR PC,SEGFERR ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT  
1973 003136 000030 30 ;*****ERROR NUMBER 30*****  
1974  
1975 003140 004737 000120 RPT6: JSR PC,0#WRTMEM ;GO TO WRITE THE MEMORY WITH A BACKGROUND OF THE  
1976 ;* WORD STORED AT LOCATION BAKPAT  
1977 003144 005037 000306 CLR 0#PASFLG  
1978 003150 010403 10: MOV R4,P3 ;SET R3  
1979 003152 010401 20: MOV R4,P1 ;AND R1 TO THE STARTING ADDRESS OF MEMORY UNDER TEST  
1980 003154 020011 30: CMP R4,(R1) ;CHECK (R1) FOR CORRECT DATA  
1981 003156 001403 REQ 40  
1982 003160 004767 JSP PC,ERROR ;*ERROR* REPORT ERROR MESSAGE  
1983 003164 000031 31 ;*****ERROR NUMBER 31*****  
1984  
1985 003166 062701 000002 40: ADD #2,R1 ;INCREMENT R1 BY 2  
1986 003172 020105 CMP R1,R5 ;SEE IF R1 HAS REACHED THE MAX. OF MEMORY  
1987 003174 103767 BLO 30 ;SEE IF R1 HAS REACHED THE MAX. OF MEMORY  
1988 003176 132737 000001 000306 RIAB #1,0#PASFLG ;CHECK TO SEE IF PASFLG#0 OR 2  
1989 003204 001002 RNE 50  
1990 003206 105237 000306 INCR 0#PASFLG ;IN WHICH CASE INCREMENT PASFLG COUNTER BY 1  
1991  
1992 003212 020305 50: CMP R3,P5 ;SEE IF R3 HAS REACHED THE MAX. OF THE MEMORY  
1993 003214 103012 BHS 70  
1994 003216 012702 037776 MOV #37776,R2 ;WRITE INTO 1 LOC FOR >2MS (ABOUT 100MS)  
1995 003222 000033 60: SWAB (R3)  
1996 003224 005302 DEC R2  
1997 003226 001375 BNE 60  
1998 003230 010337 000354 MOV R3,0#SAVLOC ;SAVE LOCATION WRITTEN FOR 2MS FOR ERROR REPORT.  
1999 003234 062703 020000 ADD #2,R0,P3 ;BY ADDING 1 TO THE 14TH ADDRESS R17 CAUSE  
 ;R3 TO POINT TO A LOCATION IN THE NEXT  
 ;4K BANK OF MEMORY  
2000  
2001  
2002 003240 000744 BR 20  
2003 003242 105237 000306 70: INCR 0#PASFLG ;MAKE PASFLG#2  
2004 003246 000337 000316 SWAR 0#BAKPAT ;IF BAKPAT IS NOT BEING SWAPPED FOR THE 2ND  
2005 003252 001732 PFQ RPT6 ;IF BAKPAT IS NOT BEING SWAPPED FOR THE 2ND  
2006 003254 000721 END6: BR E'D5 ;THEN GO BACK TO THE LOCATION RPT6
```

```
2007 ;*****  
2008 :*TEST 7 SHIFTING DIAGONAL  
2009  
2010 ;*(1) THIS TEST WRITES THE MEMORY WITH A BACKGROUND OF BAKPAT  
2011 ;*(2) IT WRITES A DIAGONAL OF SWAPPED BAKPAT THROUGH EACH MEMORY BANK  
2012 ;*(3) READS THE MEMORY FOR CORRECT DATA  
2013 ;*(4) SHIFTS THE DIAGONAL AND REPEATS STEP 3 UNTIL THE  
2014 ;* DIAGONAL HAS BEEN SHIFTED 64 TIMES  
2015 ;*(5) WRITES A BACKGROUND OF SWAPPED BAKPAT, A DIAGONAL OF  
2016 ;* BAKPAT AND REPEATS FROM STEP 3  
2017 ;*  
2018 003256 122737 000007 000404 TST7: CMPR #7,0#STESTN ;CHECK FOR PROPER TEST SEQUENCE  
2019  
2020 003264 001403 REQ ;+10  
2021 003266 004767 JSR PC,SEGFERR ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT  
2022 003272 000032 32 ;*****ERROR NUMBER 32*****  
2023  
2024 003274 005037 000306 70: CLR 0#PASFLG  
2025 003300 010337 000304 MOV R3,0#LOWBANK ;LOWBANK CONTAINS ADDRESS OF THE LOWEST LOCATION  
 ;IN THE 4K BANK THAT CAN BE TESTED  
2026  
2027 003304 010302 MOV R3,P2  
2028 003306 052702 FIS #17777,R2 ;R2 CONTAINS THE ADDRESS OF THE TOP OF THE BANK  
2029 003312 005202 INC P2 ;ADD 1 TO POINT IT TO NEXT BANK  
2030 003314 001402 PFQ 30 ;BRANCH IF ZERO (IT MUST BE A 30K SYSTEM)  
2031 003316 020502 RNE R4,R2  
2032 003320 103001 BHS 40 ;IF R2 IS GREATER THAN R5 THEN GO TO 40  
2033 003322 010502 MOV R5,P2 ;NOW R2 CONTAINS THE ADDRESS OF THE HIGHEST LOCATION  
 ;THAT CAN BE TESTED  
2034  
2035 003324 010337 000302 40: MOV R3,0#STRDI ;LOAD STRDI WITH THE STARTING ADDRESS OF THE  
 ;DIAGONAL  
2036  
2037 003330 013701 000304 MOV 0#LOWBANK,R1 ;R1 IS NOW POINTING TO THE LOWEST LOCATION IN THE 4K  
 ;BANK  
2038  
2039 003334 013700 000316 60: MOV 0#BAKPAT,R0 ;STORE THE CONTENTS OF BAKPAT IN R0  
2040 003340 020103 CMP R1,P3 ;IS R1 POINTING TO A LOCATION IN THE DIAGONAL ?  
2041 003342 001010 RNE 100 ;IF NOT THEN GO TO 100  
2042 003344 062703 000002 ADD #2,P3 ;THE FOLLOWING CODE IS USED TO PLACE THE  
2043 003350 032703 000176 RI7 #176,P3 ;ADDRESS OF THE NEXT LOCATION IN THE DIAGONAL  
2044 003354 001402 PFQ 80 ;IN R3  
2045 003356 062703 000200 ADD #200,P3 ;  
2046 003362 000300 00: SWAP R1 ;DIAGONAL WILL CONTAIN SWAPPED BACKGROUND PATTERN  
2047 003364 132737 000001 000306 100: HI7 #1,0#PASFLG ;CONTENTS OF LOCATION PASFLG WILL BE EVEN IF THE  
 ;MEMORY IS BEING WRITTEN AND IT WILL BE ODD  
 ;IF IT IS ONLY BEING READ  
 ;IF IT IS ONLY BEING READ  
 ;IF IT IS ONLY BEING READ ONLY THEN GO TO 120  
 ;OTHERWISE WRITE THE MEMORY WITH THE CONTENTS  
 ;OF R0  
2048  
2049  
2050 003372 001001 HNE 120  
2051 003374 010011 MOV R0,(P1) ;CHECK THE LOCATION POINTED BY R1 TO CONTAIN  
 ;PROPER DATA  
2052  
2053 003376 020011 120: CMP R0,(R1) ;CHECK THE LOCATION POINTED BY R1 TO CONTAIN  
 ;PROPER DATA  
2054  
2055 003400 001403 REQ 140 ;IF IT IS OK THEN GO TO 140  
2056 003402 004767 002222 JSP PC,FRR0P ;*ERROR* REPORT ERROR MESSAGE  
2057 003406 000033 33 ;*****ERROR NUMBER 33*****  
2058  
2059 003410 062701 000002 140: ADD #2,R1 ;CAUSE R1 TO POINT TO THE NEXT MEMORY LOCATION  
2060 003414 020102 CMP R1,R2 ;IS IT THE END OF THE BANK ?  
2061 003416 103746 BLO 60 ;IF NOT THEN GO TO 60  
2062 003420 005237 000410 160: INC 0#DEUCT ;TELL APT WE ARE STILL RUNNING OKAY
```

| | | | | | | | |
|------|--------|--------|--------|--------|------|-------------|--|
| 2063 | 003424 | 105237 | 000306 | | JNCR | 0#PASFLG | |
| 2064 | 003430 | 113703 | 000302 | | MOV | 0#STRDI,R3 | !LOAD R3 WITH THE STARTING ADDRESS OF THE DIAGONAL |
| 2065 | 003434 | 132737 | 000001 | 000306 | RITP | 01,0#PASFLG | !HAS THE READ OF THE MEMORY BEEN DONE ? |
| 2066 | 003442 | 001330 | | | BNE | 46 | !IF NOT THEN GO TO 46 |
| 2067 | 003444 | 005723 | | | TST | (R3)+ | !ADD 2 TO THE STARTING ADDRESS OF THE DIAGONAL |
| 2068 | 003446 | 020302 | | | CMR | R3,R2 | !AND UNLESS THE END OF THE BANK IS REACHED |
| 2069 | 003450 | 103003 | | | RHS | 1R8 | ! |
| 2070 | 003452 | 105737 | 000306 | | TSTR | 0#PASFLG | !OP THE DIAGONAL HAS BEEN ROTATED 64 TIMES |
| 2071 | 003456 | 100322 | | | RPL | 46 | !REPEAT FROM 46 |
| 2072 | 003460 | 013703 | 000304 | 1861 | MOV | 0#LOWANK,R3 | !MAKE R3 POINT TO THE LOWEST LOCATION IN THE |
| 2073 | | | | | | | !IN THE BANK UNDER TEST |
| 2074 | 003464 | 000337 | 000316 | | SWAR | 0#BAKPAT | |
| 2075 | 003470 | 001715 | | | PEO | 46 | !AND IF THE TEST HAS NOT BEEN PERFORMED WITH THE |
| 2076 | | | | | | | !SWAPPED BACK GROUND PATTERN THEN GO TO 46 |
| 2077 | 003472 | 010203 | | | MOV | R2,R3 | !MAKE THE PRESENT HIGH BOUNDARY AS THE NEXT |
| 2078 | | | | | | | !LOW BOUNDARY |
| 2079 | 003474 | 020205 | | | CMR | R2,R5 | !UNLESS THE PRESENT HIGH BOUNDARY IS ALSO THE |
| 2080 | | | | | | | !HIGH BOUNDARY FOR THE MEMORY UNDER TEST |
| 2081 | 003476 | 103676 | | | RLO | 25 | |
| 2082 | 003500 | 000665 | | | RR | E'D6 | |

| | | | | | | | | |
|------|--------|--------|--------|--------|----------|---|-------------|---|
| 2083 | | | | | !***** | | | |
| 2084 | | | | | !TEST 10 | READ RECOVERY GALLOPING TEST/EVERY 64TH CPLL | | |
| 2085 | | | | | | | | |
| 2086 | | | | | !*(1) | THIS TEST WRITES THE MEMORY WITH A BACKGROUND PATTERN | | |
| 2087 | | | | | ! | STORED AT LOCATION BAKPAT | | |
| 2088 | | | | | !*(2) | TEST BEGINS AT LOWEST LOCATION BEING TESTED | | |
| 2089 | | | | | ! | (IF IS NAME IT 'A') | | |
| 2090 | | | | | !*(3) | LETS NAME THE 1ST LOCATION IN THE ROW/COLUMN UNDER TEST AS 'B'. | | |
| 2091 | | | | | !*(4) | SWAPS BYTES FOR LOCATION 'A'. | | |
| 2092 | | | | | !*(5) | READS 'A', READS 'B' | | |
| 2093 | | | | | !*(6) | 0# = 0#*4200 (MAKES 0#*64TH CPLL I.E. 200TH OCTAL | | |
| 2094 | | | | | ! | LOCATION FROM THE PRESENT LOCATION OF 'B') | | |
| 2095 | | | | | !*(7) | REPEATS STEPS 5 AND 6 UNTIL 'B' IS GREATER THAN THE | | |
| 2096 | | | | | ! | END OF THE 4K BANK OF THE MEMORY IN WHICH 'A' IS RESIDING | | |
| 2097 | | | | | !*(8) | A = A+2 | | |
| 2098 | | | | | !*(9) | REPEATS STEPS 3-8 UNTILL 'A' REACHES THE END OF THE BANK | | |
| 2099 | | | | | !*(10) | GOES TO THE NEXT 4K BANK OF MEMORY AND REPEATS STEPS | | |
| 2100 | | | | | ! | 3-9 UNTILL THE END OF THE MEMORY | | |
| 2101 | | | | | !*(11) | AFTER EXECUTING THE TEST BYTES ARE SWAPPED AT | | |
| 2102 | | | | | ! | LOCATION BAKPAT AND STEPS 1-10 ARE REPEATED | | |
| 2103 | | | | | !*(12) | IN THIS TEST R0 IS POINTING TO LOCATION 'A', R1 TO | | |
| 2104 | | | | | ! | LOCATION 'B', R2 TO THE END OF THE 4K BANK IN WHICH THE | | |
| 2105 | | | | | ! | TEST IS TAKING PLACE AND R3 TO THE LOWEST LOCATION IN THE | | |
| 2106 | | | | | ! | COLUMN/ROW CONTAINING 'A' AND 'B' | | |
| 2107 | | | | | !*(13) | MOST OF THE CODE USED BY THIS TEST IS ALSO USED BY TEST 11 | | |
| 2108 | | | | | !***** | | | |
| 2109 | | | | | !TEST 10 | READ RECOVERY GALLOPING TEST/EVERY 64TH CPLL | | |
| 2110 | 003502 | 122737 | 000010 | 000404 | TST101 | CMR | 010,0#TESTN | !CHECK FOR PROPER TEST SEQUENCE |
| 2111 | | | | | | | | |
| 2112 | 003510 | 001403 | | | REQ | +10 | | |
| 2113 | 003512 | 004767 | 002450 | | JSR | PC,SEGEPP | | !ERROR* REPORT ERROR MESSAGE AND HALT AT FAULT |
| 2114 | 003516 | 000034 | | | | 34 | | !*****ERROR NUMBER 34***** |
| 2115 | | | | | | | | |
| 2116 | 003520 | 010402 | | | MOV | R4,R2 | | !SET R2 TO THE LOWEST MEMORY UNDER TEST |
| 2117 | 003522 | 052702 | 017776 | | RPT101 | RHS | 017776,R2 | !MAKE R2 POINT TO THE HIGHEST LOCATION IN THE 4K |
| 2118 | | | | | | | | !BANK FOR WHICH GALLOPING WILL BE PERFORMED |
| 2119 | 003526 | 062702 | 000002 | | GALLOP1 | ADD | R2,R2 | !INCREMENT R2 BY 2 |
| 2120 | 003532 | 001402 | | | | BFO | 15 | !RR IF IT WENT TO 0 (IT MUST BE A 30K SYSTEM) |
| 2121 | 003534 | 020205 | | | | CMR | R2,R5 | !IF THE HIGH BOUNDARY OF THE TEST IS HIGHER THAN |
| 2122 | 003536 | 101401 | | | | BLOS | 26 | |
| 2123 | 003540 | 010502 | | | 101 | MOV | R5,R2 | !THE MAXIMUM ALLOWED ADDRESS THEN ADJUST R2 |
| 2124 | 003542 | 005046 | | | 201 | CLR | -(SP) | |
| 2125 | 003544 | 010200 | | | | MOV | R2,R0 | |
| 2126 | 003546 | 013700 | 000316 | 401 | MOV | 0#BAKPAT,=(R0) | | !WRITE THE MEMORY UNDER TEST WITH A BACKGROUND OF |
| 2127 | | | | | | | | !BAKPAT |
| 2128 | 003552 | 020003 | | | CMR | R0,R3 | | |
| 2129 | 003554 | 101374 | | | RHI | 46 | | |
| 2130 | 003556 | 010301 | | 601 | MOV | R3,R1 | | !R3 AND R1 ARE POINTING TO THE LOWEST LOCATION THAT |
| 2131 | | | | | | | | !CAN BE TESTED IN THIS BLOCK |
| 2132 | 003560 | 023710 | 000316 | | CMR | 0#BAKPAT,(R0) | | !BEFORE STARTING THE GALLOPING TEST FOR LOCATION |
| 2133 | | | | | | | | !(R0) CHECK IT |
| 2134 | 003564 | 001410 | | | REQ | 00 | | !CONTINUE IF OK |
| 2135 | 003566 | 010001 | | | MOV | R0,R1 | | !OTHERWISE PREPARE TO REPORT THE ERROR |
| 2136 | 003570 | 013700 | 000316 | | MOV | 0#BAKPAT,R0 | | |
| 2137 | 003574 | 004767 | 002030 | | JSR | PC,ERR0P | | !ERROR* REPORT ERROR MESSAGE |
| 2138 | 003600 | 000035 | | | | 35 | | !*****ERROR NUMBER 35***** |

```

2139
2140 003602 010011 MOV R0,(R1) ;RESTORE THE CONTENTS OF (R1)
2141 003604 010100 MOV R1,R0 ;RESTORE R0
2142
2143 003606 000310 R01 SWAP (R0)
2144 003610 031011 1001 BIT (R0),(R1) ;CHECK TO SEE THAT NONE OF THE BITS SET
2145 ;IN (R0) ARE SET IN (R1) AND VICE VERSA
2146 003612 020001 CMP R0,R1 ;THE ONLY EXCEPTION TO THIS WILL BE WHEN R0=R1
2147
2148 003614 001412 REQ 120
2149 003616 021137 000316 CMP (R1),R#BAKPAT ;CHECK THAT (R1) HAS BAKPAT IN IT
2150 003622 001407 REQ 120
2151 003624 010046 MOV R0,(SP) ;SAVE R0 ON STACK
2152 003626 013700 000316 MOV R#BAKPAT,R0 ;PLACE THE PATTERN WORD IN R0
2153 003632 004767 001772 JSR PC,ERR0P ;*ERROR* REPORT ERROR MESSAGE
2154 003636 000036 36 ;*****ERROR NUMBER 36*****
2155
2156 003640 012600 MOV (SP)+,R0 ;RESTORE R0
2157 003642 021037 000320 1201 CMP (R0),R#SWAPAT ;CHECK THAT (R0) HAS SWAPPED BAKPAT IN IT
2158 003646 001412 REQ 140
2159 003650 010146 MOV R1,(SP) ;SAVE R1 ON THE STACK
2160 003652 010001 MOV R0,R1 ;MAKE R1 POINT TO THE FAILING LOCATION
2161 003654 013700 000320 MOV R#SWAPAT,R0 ;LOAD R0 WITH THE EXPECTED RESULT IN (R1)
2162 003660 004767 001744 JSR PC,ERR0R ;*ERROR* REPORT ERROR MESSAGE
2163 003664 000037 37 ;*****ERROR NUMBER 37*****
2164
2165 003666 010011 MOV R0,(R1) ;RECOVER (R1) FROM THE ERROR
2166 003670 010100 MOV R1,R0 ;RESTORE R0
2167 003672 012601 MOV (SP)+,R1 ;AND RESTORE R1
2168 003674 122737 000011 000404 1401 CmpB #11,R#TESTN ;IS THE PROGRAM EXECUTING TEST # 11 ?
2169 003702 001402 REQ 160 ;IF SO THEN GO TO 160
2170 003704 002701 000176 ADD #176,R1
2171 003710 002701 000002 1601 ADD #2,R1 ;MAKE R1 POINT TO THE NEXT ADJACENT CELL
2172 003714 020102 Cmp R1,R2 ;AND IF R1 HAS NOT REACHED THE END OF THE BOUNDRY
2173 003716 101734 RLO 100 ;THEN REPEAT FROM 100
2174 003720 000320 SWAP (R0)+ ;RESTORE THE LOCATION FOR WHICH THE GALLOPING TEST
; WAS BEING PERFORMED
2175
2176 003722 122737 000011 000404 CmpB #11,R#TESTN ;IS IT TEST 11 ?
2177 003730 001407 REQ 170 ;IF SO THEN GO TO 170
2178 003732 005723 TST (R3)+ ;OTHERWISE INCREMENT R3 BY 2
2179 003734 002716 000002 ARD #2,(SP) ;FOR EVERY ROW/COLUMN TESTED ADD 2
2180 003740 105716 TSTB (SP)
2181 003742 100002 EPL 170 ;UNTIL (SP) IS 200
2182 003744 161603 SUB (SP),R3 ;SUBTRACT 200 FROM R3
2183 003746 005016 CIR (SP)
2184 003750 032700 000177 1701 BIT #177,R0 ;AT A 64TH CALL BOUNDRY?
2185 003754 001002 RNE 100 ;BRANCH IF NO
2186 003756 005237 000410 INC R#DEVCT ;TELL APT WE ARE STILL RUNNING
2187 003762 020002 1801 Cmp R0,R2 ;IF R0 HAS NOT REACHED THE END OF THE BOUNDRY
2188 003764 103674 RLO 60 ;THEN REPEAT FROM 60
2189 003766 162603 SHR (SP)+,R3 ;RESTORE SP AND R3
2190 003770 000337 000320 SWAP R#SWAPAT
2191 003774 000337 000316 SWAB R#BAKPAT
2192 004000 001660 RFQ 20 ;IF THE LOWER BYTE OF BAKPAT IS 0 THEN REPEAT FROM 20
2193 004002 010203 MOV F2,R3 ;OTHERWISE MAKE THE PRESENT HIGH BOUNDRY AS THE
2194 ;NEXT LOW BOUNDRY
    
```

```

2195 004004 020205 Cmp F2,R5
2196 004006 001410 RFQ F#010 ;IF PREVIOUS HIGH BOUNDRY WAS THE END OF THE
2197 ;TEST BOUNDRY THEN EXIT THE TEST
2198 004010 032702 017776 BIT #1776,R2 ;WAS IT A 4K BOUNDRY ?
2199 004014 001025 RNE RPT11 ;IF NOT THEN WE WERE PERFORMING TEST 11 WITH LONG
2200 ;GALLOPING TEST DISABLED
2201 004016 122737 000011 000404 CmpB #11,R#TESTN ;IF IT IS TEST # 11 THEN GO TO REPEAT TEST 11
2202 004024 001421 RFQ RPT11
2203 004026 000635 RR RPT10 ;OTHERWISE REPEAT TEST 10
2204 004030 000623 END101 RR END7
2205
2206
2207
    
```



```

2208 ;*****
2209 ;*TEST 11 PFAD RECOVERY LONG GALLOPING/FAST GALLOPING TEST
2210
2211 ;*(1) THIS TEST WRITES MEMORY WITH BAKPAT
2212 ;*(2) THE TEST BEGINS AT THE LOWEST LOCATION BEING TESTED
2213 ;*(3) 'A', 'B' (MOVE THE ADDRESS OF 'B' TO THE POINTER FOR LOCATION 'A')
2214 ;*(4) SWAPS BYTES FOR LOCATION 'A'
2215 ;*(5) READS 'A', WRITES 'B'
2216 ;*(6) 'R'='B'+2
2217 ;*(7) IF GALLOPING OPTION BIT AT $SWREG IS HIGH THEN STEPS 4 AND 5
2218 ;*(8) ARE REPEATED UNTIL 'B' REACHES THE HIGHEST MEMORY LOCATION
2219 ;*(9) OF THE 4K BANK IN WHICH 'A' IS RESIDING, THEN 'A' IS
2220 ;*(10) INCREMENTED BY 2 AND AFTER MAKING 'B' TO POINT TO THE LOWEST
2221 ;*(11) LOCATION OF THE 4K MEMORY BANK CONTAINING 'A' STEPS 3,4,5 AND
2222 ;*(12) 6 ARE REPEATED UNTIL 'A' EQUALS THE END OF THE ENTIRE MEMORY
2223 ;*(13) IF GALLOPING OPTION BIT IS NOT HIGH THEN STEPS 4 AND 5 ARE
2224 ;*(14) REPEATED UNTIL 'R' IS POINTING TO A CELL IN THE NEXT COLUMN
2225 ;*(15) IF SEQUENTIAL CELLS LIE ALONG THE ROW, OR THE NEXT ROW
2226 ;*(16) IF SEQUENTIAL CELLS LIE ALONG THE COLUMN, AT WHICH TIME
2227 ;*(17) STEPS 2,3,4,5 AND 7 ARE REPEATED UNTIL THE END OF THE MEMORY
2228 ;*(18) TEST IS REPEATED FOR THE OPPOSITE BACKGROUND DATA
2229 ;*(19) IF THIS TEST R0 POINTS TO LOCATION 'A', R1 TO LOCATION
2230 ;*(20) 'B', R2 TO THE HIGHEST LOCATION AND R3 TO THE LOWEST
2231 ;*(21) LOCATION IN A 64/4K CELL BOUNDARY
2232 ;*(22) MOST OF THE CODE USED BY TEST 10 IS ALSO USED BY THIS TEST
2233
2234 ;*****
2235 ;*TST111 CMPH #11,#8TEST# ;CHECK FOR PROPER TEST SEQUENCE
2236 004032 122737 000011 000404
2237
2238 004040 001403 REQ #+10
2239 004042 004767 002120 JSR PC,SEQEPR ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
2240 004046 000040 40 ;*****ERRROR NUMBER 40*****
2241
2242 004050 010402 MOV R4,R2 ;*MAKE R2 TO POINT TO THE LOWEST LOCATION
2243 ;*UNDER TEST
2244 004052 105777 174372 TSTR #ASKR ;LONG GALLOP ENABLED?
2245 004056 100004 RPL RBT11 ;BRANCH IF NO
2246 004060 004767 002526 JSR PC,PNTMES ;TYPE "GLP"
2247 004064 046107 000120 ;ASCIZ /CLP/
2248 004070 105777 174354 PPT111 ;TST #ASKR ;LONG GALLOPING ENABLED?
2249 004074 100612 RMI RBT10 ;BRANCH IF YES
2250 ;*TO RPTIM
2251 004076 052702 000176 HIS #176,R2 ;*OTHERWISE SET THE LOW ORDER BITS OF THE ADDRESS
2252 ;*TO GET THE HIGH BOUNDARY
2253
2254 004102 000611 BR GALLOP ; PERFORM GALLOPING TEST

```

```

2255 ;*****
2256 ;*TEST 12 WORST CASE TESTING FOR CORE MEMORY
2257 ;*(1) STARTING FROM THE LOWEST LOCATION UNDER TEST THE MEMORY
2258 ;*(2) IS WRITTEN WITH A BACKGROUND OF BAKPAT, HOWEVER LOCATIONS
2259 ;*(3) HAVING ADDRESS SUCH THAT EXCLUSIVE OR OF ADDRESS BITS 1 &
2260 ;*(4) 8 = 1 ARE WRITTEN TO A VALUE OF SWAPPED BAKPAT
2261 ;*(5) STARTING FROM THE LOWEST LOCATION THE MEMORY IS CHECKED
2262 ;*(6) TO CONTAIN THE CORRECT DATA AS EXPLAINED IN STEPS 3 & 4,
2263 ;*(7) UNTIL THE HIGHEST LOCATION UNDER TEST IS REACHED
2264 ;*(8) READ EACH LOCATION FOR THE CORRECT CONTENT
2265 ;*(9) COMPLEMENT THE LOCATION AND READ IT) COMPLEMENT THE LOCATION
2266 ;*(10) BACK TO ITS ORIGINAL VALUE AND READ IT AGAIN
2267 ;*(11) STARTING FROM THE HIGHEST LOCATION UNDER TEST REPEAT STEPS
2268 ;*(12) 3 & 4 UNTIL THE LOWEST LOCATION UNDER TEST IS REACHED
2269 ;*(13) REPEAT STEPS 1-5, HOWEVER THIS TIME LOCATIONS WITH XOR
2270 ;*(14) OF ADDRESS BITS 8 & 13 = 1 ARE WRITTEN TO SWAPPED BAKPAT
2271 ;*(15) REPEAT STEPS 1-5, HOWEVER THIS TIME LOCATIONS WITH XOR
2272 ;*(16) OF ADDRESS BITS 3 & 9 = 1 ARE WRITTEN TO SWAPPED BAKPAT
2273 ;*(17) REPEAT STEPS 1-7 WITH A BACKGROUND OF SWAPPED BAKPAT AND
2274 ;*(18) THE LOCATIONS TO BE WRITTEN TO SWAPPED BAKPAT WRITTEN TO
2275 ;*(19) BAKPAT.
2276 ;*****
2277 ;*TST121 CMPH #12,#8TEST# ;CHECK FOR PROPER TEST SEQUENCE
2278 004104 122737 000012 000404
2279 004112 001403 REQ #+10
2280 004114 004767 002046 JSR PC,SEQEPR ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
2281 004120 000041 41 ;*****ERRROR NUMBER 41*****
2282
2283 004122 012702 000002 MOV #2,R2 ;PREPARE TO TAKE THE EXCLUSIVE OR OF ADDRESS BITS 1
2284 004126 012703 000400 MOV #40,R3 ;AND 8
2285 004132 112737 000001 MOVF #1,#PASFLG ;INITIALIZE THE COUNTER FOR THE SURTEST
2286 004140 010401 281 MOV R4,R1 ;PLACE THE STARTING ADDRESS OF MEMORY UNDER
2287 ;*TEST IN R1
2288 004142 013700 000316 MOV #BKPAT,R0
2289 004146 030201 RIT R2,R1 ;CHECK TO SEE IF ADDRESS BIT STORED IN R2 IS SET
2290 004150 001004 RNE #3 ;IF IT IS SET THEN GO TO #3
2291 004152 030301 BIT R1,R1 ;CHECK TO SEE IF ADDRESS BIT POINTED BY R3 IS SET
2292 004154 001404 BEQ 128 ;IF IT IS NOT SET THEN GO TO 128
2293 004156 005100 COM R0 ;COME HERE ONLY IF EXCLUSIVE OR OF ADDRESS BITS
2294 ;*POINTED BY R2 & POINTED BY R3 = 1 IN WHICH
2295 ;*CASE PREPARE TO WRITE THE LOCATION
2296 ;*WITH A COMPLEMENT OF LOCATIONS NOT MEETING
2297 ;*THIS CONDITION
2298 004160 000402 BR 128
2299 004162 030301 BIT R1,R1 ;COME HERE IF ADDRESS BIT POINTED BY R2 IS 1 AND

```

2300
2301 004164 001774

BEQ 65

;CHECK ADDRESS BIT POINTED BY R3
;IF ADDRESS BIT POINTED BY R3 IS 0 THEN GO TO 66

2302 004166 132737 000002 000306 1251
2303 004174 001001

R17R #2,00PASFLG
R1E 148

;IS IT 2ND OR 3RD PASS OF THE SUBTEST ?
;IF SO THEN READ THE MEMORY

```

2304 004176 010011 MOV R0,(R1) ;OTHERWISE WRITE THE MEMORY BEFORE READING IT
2305 004200 020011 148: CMP R0,(R1) ;READ THE MEMORY FOR CORRECT CONTENTS
2306 004202 001403 REQ 168
2307 004204 004767 001420 JSR PC,ERRPOR ;*ERROR* REPORT ERROR MESSAGE
2308 004210 000042 42 ;*****ERROR NUMBER 42*****
2309
2310 004212 012786 000002 168: MOV #2,-(SP)
2311 004216 005100 188: COM R0
2312 004220 005111 COM (R1)
2313 004222 020011 CMP R0,(R1) ;READ THE MEMORY AGAIN
2314 004224 001404 BFO 198
2315 004226 004767 001376 JSR PC,ERRPOR ;*ERROR* REPORT ERROR MESSAGE
2316 004232 000043 43 ;*****ERROR NUMBER 43*****
2317
2318 004234 010011 MOV R0,(R1) ;RESTORE THE LOCATION (R1)
2319 004236 005316 198: DEC (SP)
2320 004240 001366 BNE 188 ;EXECUTE THE CODE FROM 188 TWICE
2321 004242 005726 TST (R0)+ ;RESTORE THE STACK POINTER
2322 004244 127737 000003 000306 CMPB #3,0#PASFLG ;IS IT THE 3RD PASS OF THE SUBTEST ?
2323 004252 001412 REQ 208 ;IF SO THEN GO TO 208
2324 004254 062701 000002 ADD #2,R1 ;IN FIRST 2 PASSES THE PROGRAM PROCEEDS IN
2325 ;MIN. TO MAX. DIRECTION
2326 004260 020105 CMP R1,R5 ;HAVE WE REACHED THE MAX. ADDRESS UNDER TEST ?
2327 004262 103727 RLO 48 ;IF NOT THEN REPEAT FROM 48
2328 004264 105237 000306 INCB 0#PASFLG
2329 004270 127737 000002 000306 CMPB #2,0#PASFLG ;IF IT IS THE 2ND PASS OF THE SUBTEST
2330 004276 001720 BEQ 26 ;THEN REPEAT FROM 26
2331 004300 162701 000002 208: SUB #2,R1 ;OTHERWISE EXECUTE THE TEST IN MAX. TO MIN.
2332 ;DIRECTION
2333 004304 020104 CMP R1,R4 ;HAVE WE REACHED THE MIN. ADDRESS UNDER TEST ?
2334 004306 103315 BHIS 48 ;IF NOT THEN REPEAT FROM 48
2335 004310 012702 020000 MOV #20000,R2 ;PREPARE TO CHECK THE MEMORY WITH THE XOR OF
2336 ;ADDRESS BITS 8 AND 13
2337 004314 105237 000307 INCB 0#PASFLG+1 ;THE SUB TEST HAS CHECKED THE XOR ONE KIND
2338 004320 123727 000307 000002 CMPB 0#PASFLG+1,#2 ;HAS TWO XOR COMBINATIONS BEEN CHECKED ?
2339 004326 103701 RLO 18 ;IF NOT THEN GO TO 18
2340 004330 101004 BHI 228 ;IF ALL THREE HAVE BEEN CHECKED THEN GO TO 228
2341 004332 012702 000010 MOV #10,R2 ;IF IT IS THE 2ND XOR COMBINATION THEN CHECK
2342 004336 006303 ASL R3 ;FOR ADDRESS BITS 3 & 8
2343 004340 000674 BR 18
2344 004342 005137 000316 228: COM 0#RAKPAT ;IF THE TEST WAS NOT PERFORMED WITH THE SWAPPED
2345 004346 105737 000316 TSTB 0#RAKPAT
2346 004352 001684 REQ TST12 ;BAKPAT THEN RE-EXECUTE THE TEST
2347 004354 000625 END121 BR E1D10
    
```

```

2348 ;*****
2349 ;*TEST 13 WRITE RECOVERY TEST
2350 ;* THIS TEST DIFFERS FROM 0-12 IN THAT IT CONSISTS OF A SMALL TEST PROGRAM
2351 ;* ACTUALLY RUNNING IN THE 4K BANK UNDER TEST.
2352 ;* THE PROGRAM IS SELF MODIFYING AND MAY BE DIFFICULT TO DEBUG.
2353 ;* TO AID IN THE DEBUG, BEFORE A BANK IS ENTERED "TST13 ROK XX"
2354 ;* IS TYPED. THIS WILL ALLOW THE USER TO AT LEAST SEE WHICH MEMORY
2355 ;* BANK FAILED.
2356 ;* THE TEST CONSISTS OF 1/2 OF THE BANK STORED WITH "MOV R2,-(PC)"
2357 ;* AND THE OTHER 1/2 CONTAINING "177667". "177667" IS THE COMPLEMENT
2358 ;* OF "JMP (R0)" INSTRUCTION.
2359 ;* R2 CONTAINS "COM -(R1)" INSTRUCTION ON ENTRY TO THE BANK AND R1 CONTAINS
2360 ;* THE HIGHEST TEST ADDRESS IN THAT BANK. THE HIGHEST TEST ADDRESS IS
2361 ;* USUALLY ON 4K BOUNDARIES, WHEN TESTING BANK 0 RELOCATED, HOWEVER
2362 ;* R1 CONTAINS THE FIRST FREE TEST ADDRESS BELOW THE DIAGNOSTIC.
2363 ;* IF YOU UNDERSTAND THIS SO FAR THE REST IS EASY.
2364 ;* THE TEST EXECUTION IS AS FOLLOWS:
2365 ;* 1. THE "MOV R2,-(PC)" INSTRUCTION EXECUTES STOPPING
2366 ;* THE CONTENTS OF R2 IN THE ADDRESS IT VACATED (DUE TO -(PC).
2367 ;* 2. SINCE R2 CONTAINS A "COM -(R1)" INSTRUCTION IT COMPLEMENTS
2368 ;* THE HIGHEST ADDRESS UNDER TEST. THIS ADDRESS CONTAINED
2369 ;* "177667" SO AFTER THE COM -(R1) IT EQUALS 110
2370 ;* CLEVERLY THIS IS THE "JMP (R0)" INSTRUCTION.
2371 ;* 3. THIS SEQUENCE CONTINUES UNTIL THE "MOV R2,-(PC)" INSTRUCTIONS
2372 ;* REACH THE MIDDLE OF THE TEST BANK. THEN THE "JMP (R0)" INSTRUCTION IS
2373 ;* AND EXECUTED. R0 CONTAINED THE RETURN ADDRESS BACK
2374 ;* TO TEST 13.
2375 ;* 4. THESE STEPS ARE REPEATED FOR EACH BANK UNDER TEST.
2376 ;*
2377 ;*****
2378 004356 122737 000013 000404 TST131 CMPB #13,0#TESTN ;CHECK FOR PROPER TEST SEQUENCE
2379 004364 001403 BFO #10
2380 004366 004767 001574 JSR PC,SEGERP ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
2381 004372 000044 44 ;*****ERROR NUMBER 44*****
2382
2383 004374 012702 010247 181: MOV #10247,R2 ;PLACE THE OP CODE OF INSTRUCTION MOV R2,-(PC)
2384 ;IN R2
2385 004400 012700 177667 MOV #177667,R0 ;PLACE THE COMPLEMENT OF THE INSTRUCTION
2386 ;JMP (R0) IN R0
2387 ;INSURE LOWEST TEST ADDRESS TO END OF 4K SEGMENT IS MULTIPLE OF 2
2388 ;SINCE THE TEST STORES "MOV R2,-(PC) IN 1/2 AND 177667 IN THE OTHER 1/2
2389
2390 004404 010546 281: MOV R5,-(SP) ;SAVE R5
2391 004406 010446 MOV R4,-(SP) ;STORE LOWEST ADDRESS ON STACK
2392
2393 004410 000241 298: CLC ;MAKE POSITIVE BYTE COUNT OF HIGH ADDRESS
2394 004412 006005 ROR R5 ;DO SAME FOR LOWEST ADDRESS
2395 004414 006004 ROR R4 ;GET DIFFERENCE OF LOWEST ADDRESS AND HIGHEST
2396 004416 100405 SUB R4,R5 ;IF DIFFERENCE IS ODD THEN R4 IS AT LOWEST ADDRESS
2397 004422 103002 ROR R5 ;BRANCH IF R4 IS AT LOWEST TEST ADDRESS.
2398 004424 062716 000002 ADD #2,(SP) ;INCREASE LOWEST TEST ADDRESS BY 2
2399 004430 012604 308: MOV (SP)+,R4 ;RESTORE R4 (POSSIBLY INCREASED BY 2 FROM ENTRY)
2400 004432 012605 MOV (SP)+,R5 ;RESTORE HIGHEST TEST ADDRESS
2401 004434 010403 MOV R4,R3 ;PLACE THE LOWEST LOCATION UNDER TEST
2402 ;IN R3
2403 004436 000406 BR 288 ;LEAVE LOW BITS OF R3 ALONE FIRST TIME IN CASE BANK 0
    
```

```

2404 004440 042703 017776 30: BIC #17776,R3 ;CAUSE R3 TO POINT TO THE LOWEST LOCATION
2405 ;IN THE 4K BANK UNDER TEST
2406 004444 001507 HEQ 148 ;IF ADDRESS WENT TO 0, IT MUST BE A 30K SYSTEM
2407 004446 105737 000405 TSTB #RFL ;ARE WE RELOCATED?
2408 004452 100504 RMI 148 ;BRANCH IF YES-TEST BANK ONLY-
2409 004454 020305 280: CMP R3,R5 ;IF R3 IS HIGHER THAN THE HIGHEST LOCATION
2410 004456 103102 BHIS 148 ;UNDER TEST THEN EXIT
2411 ;IF R5 LESS THAN 20000 THEN WE ARE TESTING BANK0 RELOCATED IN BANK0
2412 004460 020527 020000 CMP R5,#20000 ;IS HIGHEST TEST ADDRESS BELOW 4K?
2413 004464 103002 RHIS 318 ;BRANCH IF NO
2414 004466 010501 MOV R5,R1 ;SET R1 TO HIGHEST TEST ADDRESS IN BANK0
2415 004470 000407 BR 328
2416
2417 004472 010301 318: MOV R3,R1 ;SET R1 TO LOWEST CURRENT TEST ADDRESS
2418 004474 052701 017777 RIG #17777,R1 ;SET LOW ORDER ADDRESS BITS
2419 004500 005201 INC R1 ;CAUSE R1 TO POINT TO THE HIGHEST LOCATION+2
2420 ;OF THE 4K BANK BEING POINTED BY R3
2421 004502 001402 REG 328 ;BRANCH IF R1 WENT TO 0 (WHICH MIGHT
2422 ;HAVE HAPPENED IF TESTING A 30K LSI SYSTEM)
2423 004504 020105 CMP R1,R5 ;COMPARE R1 TO HIGHEST ADDRESS UNDER TEST
2424 004506 101401 BLOS 338 ;BRANCH IF WITHIN RANGE
2425 004510 010501 320: MOV R5,R1 ;SET R1 TO THE MAXIMUM AVAILABLE MEMORY
2426
2427 004512 132737 000001 000306 338: RITB #1,#PASFLG ;IS THE LOWEST BIT OF LOCATION PASFLG
2428 004520 001101 RNE 168 ;SET? IN WHICH CASE BACK GROUND HAS
2429 ;ALREADY BEEN WRITTEN AND WRITE RECOVERY
2430 ;TEST IS BEING PERFORMED
2431
2432 004522 020304 48: CMP R3,R4 ;OTHERWISE WRITE THE BACKGROUND
2433 004524 103430 RLO 88 ;DEFINED AT STEP 3.
2434 004526 105737 000307 TSTB #PASFLG+1 ;IS THE TEST JUST DOING READ, I.E.
2435 004532 001002 RNE 68 ;IS THE PASFLG+1 LOCATION NON ZERO? IF SO
2436 ;THEN GO TO 68
2437 004534 012713 010247 MOV #10247,(R3) ;WRITE THE LOCATION WITH THE OP CODE FOR MOV R2,=(PC)
2438 004540 020213 68: CMP R2,(R3) ;READ (R3) TO CONTAIN CORRECT DATA
2439 004542 001421 BEQ 88
2440 004544 010046 MOV R0,-(SP) ;SAVE R0
2441 004546 010146 MOV R1,-(SP) ;AND R1 ON THE STACK
2442 004550 010301 MOV R3,R1
2443 004552 010200 MOV R7,R0 ;SET R0= GOOD DATA FOR ERROR PRINTOUT
2444 004554 004767 001050 JSP PC,ERR0P ;*ERROR* REPORT ERROR MESSAGE
2445 004560 000045 45 ;*****ERROR NUMBER 45*****
2446
2447 004562 012601 MOV (SP)+,R1 ;PSTORE P1
2448 004564 012600 MOV (SP)+,R0 ;AND R0
2449 004566 105737 000306 TSTB #PASFLG ;IF PASFLG IS 0 AND THE MEMORY DOES NOT HAVE
2450 ;THE PROPER DATA THEN WE DON'T WANT TO GO AND
2451 ;EXECUTE THE INSTRUCTIONS STORED IN MEMORY UNDER
2452 ;TEST
2453 004572 001005 RNE 88 ;BRANCH IF PASFLG NOT =0
2454
2455 004574 010200 MOV R7,R0 ;SAVE FOR ERROR REPORT
2456 004576 004767 001026 JSP PC,ERR0R ;*ERROR* REPORT ERROR MESSAGE
2457 004602 000046 46 ;*****ERROR NUMBER 46*****
2458
2459 004604 000663 BP #D12 ;ABORT TEST 13.
    
```

```

2460
2461 004606 062703 000002 88: ADD #2,R3 ;INCREMENT R3 BY 2
2462 004612 162701 000002 SUB #2,R1 ;DECREMENT R1 BY 2
2463 004616 020105 CMP R1,R5 ;WRITE THE BACKGROUND DEFINED AT STEP 4.
2464 004620 103014 BHIS 178
2465 004622 020103 CMP R1,R3 ;HAS STORING THE 177667 REACHED WHERE *MOV R2,=(PC) IS?
2466 004624 103405 ELO 148 ;BRANCH IF YES DON'T DESTROY THE MOV R2,=(PC) IS.
2467 004626 105737 000307 TSTB #PASFLG+1 ;IS THE THE READ ONLY CHECK PASS?
2468 004632 001002 RNE 148 ;BRANCH IF YES
2469 004634 012711 177667 MOV #177667,(R1) ;WRITE THE LOCATION WITH THE COMPLEMENT OF THE
2470 ;OP CODE JMP (R0)
2471 004640 020011 108: CMP R0,(R1) ;READ R1 TO CONTAIN CORRECT DATA
2472 004642 001403 BEQ 178
2473 004644 004767 000760 JSP PC,FPP0P ;*ERROR* REPORT ERR0P MESSAGE
2474 004650 000047 47 ;*****ERROR NUMBER 47*****
2475
2476 004652 020301 128: CMP R1,R1 ;IF WE HAVE NOT REACHED THE MIDDLE OF 4K BANK
2477 004654 103722 RLO 48 ;THEN REPEAT FROM 48
2478
2479 ;RETURN HERE AFTER PROGRAM RUN IN BANK UNDER TEST
2480
2481 004656 062703 020000 136: ADD #20000,P3 ;OTHERWISE GO TO THE NEXT 4K BANK
2482 004662 000666 BR 36
2483
2484 004664 122737 000001 000306 148: CYPB #1,#PASFLG ;THE PROGRAM CONTROL COMES HERE AS FOLLOWS
2485 ;1-PASFLG=0, PROGRAM HAS JUST COMPLETED A
2486 ; WRITE/READ CYCLE FOR THE BACK GROUND
2487 ; AND WANTS TO BEGIN THE WRITE RECOVERY TEST
2488 004672 001437 BEQ 248 ;2-PASFLG=1, PROGRAM HAS JUST COMPLETED
2489 ; THE WRITE RECOVERY TEST AND WANTS TO
2490 ; READ MEMORY FOR CORRECT DATA
2491 004674 103627 RLO #D12 ;3-PASFLG=2, PROGRAM HAS CORRECTLY READ THE
2492 ; MEMORY AND WANTS TO GO THE NEXT TEST.
2493
2494 004676 105137 000307 COMB #PASFLG+1 ;ENTER HERE WITH PASFLG=0, ON THE FIRST ENTRY
2495 ;ENABLE READ ONLY FOR THE MEMORY AND ON THE SECOND
2496 ;ENTRY DISABLE READ ONLY
2497 004702 001240 RNE 28
2498 004704 012702 005141 MOV #5141,R2 ;PLACE THE OP CODE FOR INSTRUCTION COM -(R1)
2499 ;IN R2
2500 004710 012700 177740 MOV #138-16,R0 ;PLACE THE RETURN ADDRESS IN R0 AS 138
2501 004714 060700 ADD PC,R0 ;THUS WHEN THE READ RECOVERY TEST REACHES
2502 ;THE MIDDLE OF THE 4K MEMORY THEN THE
2503 ;INSTRUCTION EXECUTED WILL BE JMP (R0)
2504 ;BRANCHING THE PROGRAM TO 138
2505 004716 105237 000306 158: INCB #PASFLG ;INCREMENT PASFLG BY 1.
2506 004722 000630 BP 28
2507
2508 004724 032777 000020 173516 168: BIT #20,#SWR ;HAS THE PRINTOUTS BEEN SUPPRESSED ?
2509 004732 001016 RNE 188 ;IF SO THEN GO TO 188
2510 004734 105737 000042 TSTB #42 ;IS THE PROGRAM RUNNING UNDER ACT?
2511 004740 001013 RNE 188 ;BRANCH IF YES
2512 004742 004767 001644 JSR PC,PNTMES ;TYPE THE BANK UNDER TEST
2513 004746 051524 030524 020063 ,ASCIZ /TST13 BNK/
2514 004754 047102 000113 ,EVEN
2515
    
```

```

2516 004760 004767 002454      JSP  PC,GETRANK ;GET RANK NO. UNDER TEST INTO DECWRD FOR PRINT.
2517 004764 004767 001650      JSR  PC,$TPDEC  ;TYPE RANK NO. UNDER TEST
2518
2519 004770 000113              1881 JMP  (R3)      ;BEGIN EXECUTING MOV R2,=(PC) ,COM =(R1) SEQUENCE IN TES
2520
2521
2522 004772 105137 000307      2481 COMP  R#PASFLG+1 ;PLACE THE OP CODE FOR JMP (R0) IN R0
2523 004776 012700 000110      MOV  #110,R0    ; READ THE MEMORY FOR CORRECT DATA AFTER
2524 005002 000745              BP    158       ; INCREMENTING PASFLG TO 2
2525
2526
2527 ;TST13 EXITS VIA END12.
2528
  
```

```

2529 005004 012737 000377 000316 RELOC: MOV  #377,R#BAKPAT ;IS THE MEMORY MANAGEMENT BEING TESTED ?
2530 005012 105737 000276      TSTB R#MVA     ;IF SO THEN GO TO CONTMM AND CONTINUE TESTING
2531 005016 001065              BNE  CONTMM    ;MEMORY MANAGEMENT
2532                               ;RELOCATION WANTED?
2533 005020 032777 001000 173422  BIT  #1000,05WR ;RELOCATION WANTED?
2534 005026 001046              RNE  CKDONE    ;BRANCH IF NO
2535 005030 105737 000405      TSTB R#RFL     ;IF THE PROGRAM HAS ALREADY BEEN RELOCATED THEN ALSO
2536 005034 100420              BMI  PLOPER    ; PLACE THE PROGRAM RACK IN LOWER CORE
2537 005036 112737 000200 000405  MOVR #200,R#PEI ;OTHERWISE PREPARE TO RELOCATE
2538
2539 ;RELOCATE THE DIAGNOSTIC TO HIGHEST AVAILABLE MEMORY
2540
2541
2542 005044 004767 001542      JSP  PC,PNTMES ;TYPE "REL"
2543 005050 042522 047514 000103  .ASCIZ /RFLOC/
2544                               .EVEN
2545 005056 013705 000340      MOV  R#MAXMEM,P5 ;PREPARE TO LOAD THE PROGRAM IN THE HIGHEST
2546                               ;AVAILABLE MEMORY
2547 005062 014445              281  MOV  =(R4),=(R5) ;RELOCATE THE PROGRAM
2548 005064 020427 000430      CMP  R4,#RFGIN-50 ;NEITHER RELOCATE NOR TEST LOCATIONS LOWER THAN BEGIN=50
2549 005070 011374              RHI  28
2550 005072 000165 000050      JMP  50(R5)
2551
2552 ;*RELOCATE THE DIAGNOSTIC BACK TO LOWER MEMORY
2553
2554
2555 005076 013705 000346      RELOC: MOV  R#SAVR5,R5 ;STORE R5
2556 005102 105737 000405      TSTB R#RFL     ;IS DIAGNOSTIC IN RELOCATED STATE?
2557 005106 100016              RPL  CKDONE    ;BRANCH IF NO
2558
2559
2560 005110 012704 000430      281  MOV  #BEGIN=50,R4 ;PREPARE TO RELOCATE THE PROGRAM TO LOWER CORE
2561 005114 012524              MOV  (R5)+(R4),
2562 005116 020537 000340      CMP  R5,#MAXMEM
2563 005122 013774              BLO  28
2564 005124 105037 000405      CLR  R#REL
2565 005130 010537 000346      MOV  R5,R#SAVR5 ;SAVE R5
2566 005134 012706 000500      MOV  #BEGIN,SP  ;RESET STACK TO LOWER MEMORY
2567 005140 010637 000350      MOV  R#SAVR6   ;"BEGIN" USES THIS TO RESET THE STACK.
2568 005144 000137 005150      CKDONE: JMP  R#LOWER ;TRANSFER THE PROGRAM CONTROL TO THE LOWER CORE
2569
2570
2571 005150 105737 000315      LOWEP: TSTB R#SAVKB ;HERE DUE TO "C TYPED?
2572 005154 001073              BNE  $TPSK     ;BRANCH IF YES (TYPE ERROR STACK)
2573 005156 004767 001702      TSTMM: JSR  PC,MEMMNG ; SET THE REGISTERS IF THE MEMORY MANAGEMENT
2574                               ;IS AVAILABLE
2575 005162 105737 000276      TSTB R#MVA     ;IS MEM. MANAG. AVAILABLE ?
2576 005166 001462              BR  EDPAS     ;BRANCH IF NO
2577 005170 000402              BQ  $CNTMM    ;BEGIN TESTING ABOVE 28K
2578 005172 004767 002036      CONTMM: JSP  PC,UPMM ;GO TO UPDATE MEM. MANAG. REGISTERS
2579 005176 012703 000324      $CNTMM: MOV  #LOWWO,R3 ;MAKE R3 POINT TO THE LOCATION LOWWO
2580 005202 004767 002142      JSR  PC,GETSIZ ; LOAD BITS 6-10 OF R2 WITH THE BITS 13-17
2581                               ;OF THE LOWEST ADDRESS UNDER TEST
2582 005206 012704 020000      MOV  #2000,R4  ;MAKE R4 POINT TO THE LOWEST LOCATION IN THE BANK
2583                               ;POINTED BY PAGE ADDRESS REGISTER 1 (PAR1)
2584 005212 020237 172342      CMP  R2,#172342 ;IS THE CONTENT OF R2 LOWER THAN THE CONTENT OF
  
```

```

2585 ;PAR1 7
2586 005216 103405 BLO 28 ;IF SO THEN GO 28
2587 005220 050104 BIS R1,R4 ;SUBROUTINE GETSIZ LOADED R1 WITH BITS 0-12
2588 ;OF LOWADD WHICH HAVE NOW BEEN LOADED IN R4
2589 005222 162702 000200 SUB #20,R2 ;SET MEM. MANAG. REGISTERS
2590 005226 004767 001636 JSR PC,MWREG ;PLACE BITS 13-17 OF HIGHEST LOCATION TO BE TESTED
2591 005232 004767 002112 261 JSR PC,GETSIZ ;IN BITS 6-10 OF R2, #160000 IN R0 AND BITS 0-12
2592 ;OF LOCATION HIGHADD IN R1
2593 ;GET THE ADDRESS OF MAX. MEM. UNDER TEST
2594 005236 004767 000200 JSR PC,MAXADR
2595 005242 010005 MOV PC,P5
2596 005244 004767 002100 JSR PC,GETSIZ ;PREPARE TO SET UP LOCATION MAXMEM
2597 005250 004767 000006 JSR PC,MAXADR ;GET THE MAXIMUM ADDRESS OF AVAILABLE MEMORY
2598 005254 010013 MOV R4,(R3) ;AND STORE INTO "MAXMEM"
2599 005256 000167 174256 JMP CLRMEM ;GO TEST A 24K SLICE ABOVE 28K.
2600
2601 ;MAXADR = SUBROUTINE TO GET CURRENT 24K SLICE OF MEMORY ADDRESSES ABOVE 28K.
2602 ;REGISTERS:
2603 ;R0= ON ENTRY= #160000 AND ON EXIT=HIGHEST VIRTUAL ADDR. UNDER TEST
2604 ;R1= LOW ORDER 12 BITS OF VIRTUAL TEST ADDRESS
2605 ;R2= PAR BLOCK NO. CURRENTLY UNDER TEST.
2606
2607 MAXADR: MOV R4,-(SP) ;PUT MAXIMUM AVAILABLE ADDRESS ON STACK
2608 005262 010046 172356 MOV #172356,R0 ;R0=PAR7 UNIBUS ADDRESS
2609 005264 012700 172356 ;**BEGIN LOOP TO FIND PAR ADDRESS UNDER TEST
2610
2611 005270 162716 020000 261 SUB #2000,(SP) ;DECREMENT VIRTUAL ADDRESS BY 4K
2612 005274 050116 R1,(SP) ;SET BITS 11:0 TO MAXIMUM VIRTUAL TEST ADDRESS
2613 005276 020240 CMP R2,(R0) ;DOES CURRENT PAR= TEST BLOCK NO.?
2614 005300 001410 BEQ 36 ;BRANCH IF YES
2615 005302 020027 172340 CMP R0,#172340 ;ARE WE AT PAR#?
2616 005306 101370 HHI 25 ;NO KEEP TRYING
2617 ;**END LOOP TO FIND PAR ADDRESS UNDER TEST
2618 005310 005720 TST (R0)+ ;SET TO PAR CURRENT
2619 005312 021002 CMP (R0),R2 ;IS THE PAR BLOCK UNDER TEST GTR THAN ALLOWED?
2620 005314 003006 RGT 48 ;BRANCH IF YES (FALL INTO ENDPAS)
2621 005316 012716 157776 MOV #157776,(SP) ;FIXIT WITH MAXADR= 28K SEGMENT TEST SIZE
2622 005322 012600 381 MOV (SP)+,R0 ;SET R0 TO MAXIMUM VIRTUAL TEST ADDRESS
2623 005324 062700 ADD #7,R0 ;MAKE MAXIMUM MEMORY+2
2624 005330 000207 RTS ;AND EXIT MAXADR ROUTINE
2625
2626 005332 022626 481 CMP (SP)+,(SP)+ ;FIXUP STACK
2627 ;AND FALL THRU TO ENDPAS.

```

```

2628 ;* TYPE ROUTINE FOR ERROR STACK
2629 ;* -----
2630 ;*
2631 ;* THIS ROUTINE IS USED TO DETERMINE IF TYPE OUT OF THE ERROR STACK
2632 ;* FOR ONLY THE FAILING BITS IS REQUIRED OR NOT
2633
2634
2635
2636 005334 032777 000200 173106 ENDPAS: BIT #20,R500 ;ARE WE GOING TO TYPE THE ERROR STACK AND END OF PASS?
2637 005342 001055 BFE 500P ;IF NOT THEN GO TO 500P
2638 005344 012746 177777 2638 STPSTK: MOV #1,-(SP) ;THE PROGRAM HAS REACHED THE END AND EPROM
2639 ;STACK AND END OF PASS WILL BE TYPED OUT
2640 005350 012701 007744 MOV #ENDPPG,R1 ;PLACE THE STARTING ADDRESS OF THE ERROR STACK
2641 ;FOR 0 TO 4K MEMORY IN R1
2642 005354 012703 000376 TYPSTK: MOV #376,R3
2643 005360 005216 INC (SP) ;IF WE HAVE GONE THRU THE ENTIRE
2644 005362 020137 000310 CMP R1,#ENDSTK ;HAS THE END OF THE ERROR STACK BEEN REACHED?
2645 005366 103043 BHS 500P ;IF YES GO TO TYPE END OF PASS
2646 005370 112702 000022 MOV #R0,R2
2647 005374 105302 RETSTK: DECR R2 ;IF ALL 16 BITS OF THIS BANK HAVE BEEN CHECKED.
2648 005376 002766 BIT TYPSTK ;BANK CHECKED FOR ERROR THEN SEE IF THERE
2649 ;IS ANY MORE 4K MEMORY BANK
2650 005400 105721 TSTR (R1)+ ;OTHERWISE CHECK THE BYTE STORED AT (R1)
2651 005402 001774 BFTSTK ;IF IT IS 0 WE WILL NOT TYPE IT
2652 005404 020227 000020 CMP R2,#16. ;IS THE POINTER POINTING TO ERROR STACK BYTE
2653 ;MEANT FOR COLLECTING ADDRESS FAILURES FOR
2654 ;THE SPECIFIC MEMORY BANK
2655 005410 103404 BLO 28 ;IF NOT THEN GO TO TYPE BIT NUMBER
2656 005412 101026 HHI PFFPL ;IF IT IS POINTING TO THE STACK LOCATION INTENDED
2657 ;TO COLLECT PARITY FAILURES THEN GO TO PARFL
2658 005414 004767 001000 JSR PC,TPADR ;OTHERWISE TYPE "ADDRESS ERROR"
2659 005420 000404 BR FAILNM
2660 005422 010237 000312 261 MOV R2,#DECWRD ;PREPARE TO TYPE THE NUMBER OF THE FAILING BIT
2661 ;IN DECIMAL
2662 005426 004767 001202 JSR PC,TYPDEC ;GO TO TYPE THE BIT NUMBER IN DECIMAL
2663 005432 011637 000312 FAILNM: MOV (SP),#DECWRD ;PREPARE TO TYPE THE PAGE NUMBER
2664 005436 004767 001176 JSR PC,STPDEC ;IN DECIMAL
2665 005442 005043 CLR -(R3)
2666 005444 114113 MOV# -(R1),(R3) ;PREPARE TO PRINTOUT THE NUMBER OF TIMES THIS
2667 ;FAILURE OCCURED
2668 005446 105021 CLR# (R1)+ ;CLEAR THE ERROR STACK
2669 005450 005043 CLR -(R3)
2670 005452 105237 000314 INCB #4,TYPCNT ;ENABLE THE TYPE OUT OF 1 WORDS
2671 005456 004767 001316 JSR PC,RPPTOCT ;TYPE THE 4K BANK AND THE NUMBER OF TIMES
2672 ;THIS FAILURE WAS SEEN
2673 005462 012703 000376 MOV #376,R3 ;RESET SCRATCH STACK FOR EACH BIT PRINTED.
2674 005466 000742 BR PFTSTK
2675 005470 004767 000750 PARFL: JSR PC,TPPRER ;TYPE "PAR ERR"
2676 005474 000756 BR FAILNM

```

```

2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688 005476 005002 002014 $EOP: CLP P2 ;SET P2= PARITY MODULE DISABLE CODE
2689 005500 004767 000315 JSP PC,PARITY ;GO DISABLE PARITY MODULES IF SELECTED.
2690 005504 105737 000315 TSTR R0,SAVKBR ;CONTROL-C TYPED?
2691 005510 001043 RNE CTLC ;BRANCH IF YES=RESTORE LOADERS AND HALT=
2692 005512 005237 000406 INC R0,SPASS ;INCREMENT PASS COUNT
2693 005516 032777 000040 172724 BIT #4,R0,SWR ;"PASS*XX" PRINTOUT WANTED?
2694 005524 001012 BNE ACT11 ;BRANCH IF NO
2695 005526 004767 001052 TYPEOP: JSR PC,TPCLRF ; TYPE CR, LF, AND "PASS#"
2696 005532 040520 051523 000043 .ASCIZ /PASS#
2697
2698 005540 013717 000406 000312 .EVEN
2699 005546 004767 001066 MOV R0,SPASS,R0,DECRD ;GET PASS COUNT
2700 005552 013700 000042 JSR PC,STPDEC ;TYPE IT
2701 005556 001405 BEQ $DOAGN ;GET THE MONITOR ADDRESS
2702 005560 004767 000012 JSR PC,FLODER ;IF NONE
2703 005564 000005 RFSRT ;RESTORE XXDP MONITOR
2704 ;RETURN TO ACT11 MONITOR.
2705
2706
2707 005566 000137 000156 ;* SERVICE XXDP/ACT11
2708 JMP R0,SENCAD ;JUMP TO ACT SERVICE
2709
2710 005572 000137 000250 $DOAGN: JMP R0,RESTRT ;REPEAT TEST IF NOT UNDER ACT11/XXDP
2711
2712 005576 004767 001616 FLODER: JSR PC,CLPMM ;STOP MEMORY MANAGEMENT SO CAN RESTORE LOADERS
2713 005602 013704 000344 MOV R0,SAVR4,R4 ;RESTORE R4 WITH SAVR4
2714 005606 014445 48: MOV R0,(R4),(R5) ;RESTORE LOADERS
2715 005610 020437 000310 CMP R4,R0,ENDSTK
2716 005614 011374 RHI 48
2717 005616 000207 RTS PC ;RETURN FROM RLODER CALL
2718
2719 ;CONTROL C HANDLER
2720 005620 004767 177752 CTLC: JSP PC,RLODER ;RESTORE ABS LOADER
2721 005624 000167 000400 JMP ADTHLT ;IF NOT APT HALT AT FATHLT
2722
2723
    
```

```

2724
2725
2726
2727
2728
2729
2730
2731 005630 017617 000000 000402 ERRPR: MOV R0,(SP),R0,FATAL ;LOAD THE LOCATION $FATAL WITH THE ERROR NUMBER
2732 005636 010346 18: MOV R3,=(SP) ;SAVE R3
2733 005640 010046 MOV R0,=(SP) ;AND R0 ON THE STACK
2734
2735 ;SETUP BANK NO. IN $FATAL FOR APT
2736
2737 005642 010103 MOV R1,R3 ;GET VIRTUAL ADDRESS UNDER TEST FOR GETBNK
2738 005644 004767 001570 JSR PC,GETBNK ;GET BANK NO. UNDER TEST INTO BNK
2739 005650 013703 000312 MOV R0,=BNK,R3 ;GET BANK UNDER TEST
2740 005654 110337 000403 MOVR R3,R0,$FATAL+1 ;STORE FAILING BANK NO. FOR APT
2741
2742
2743
2744 005660 010346 MOV R3,=(SP) ;TEMPORARILY STORE R3
2745 005662 012703 000376 MOV R3,R3 ;MAKE R3 AS THE STACK POINTER
2746 005666 013743 000306 MOV R0,=PASFLG,=(R3) ;OUTPUT THE WORD STORED AT
2747 005672 005043 26: CLR R0 ;R0
2748 005674 113713 000402 MOV R0,=$FATAL,(R3) ;PUT ERROR NO. ON ERROR STACK
2749 005700 016643 000006 MOV R0,(SP),=(R3) ;PLACE THE RETURN PC AT (R3)
2750 005704 011143 MOV R0,(R1),(R3) ;PLACE BAD DATA,
2751 005706 010043 MOV R0,=(R3) ;AND GOOD DATA ON THE STACK
2752 005710 005043 CLR R0 ;R0
2753 005712 016313 000004 MOV R0,4(R3),(R3) ;TAKE THE
2754 005716 040013 BIC R0,(R3) ;EXCLUSIVE OR OF GOOD AND BAD DATA
2755 005720 046300 000004 RIC 4(R3),R0 ;TO FIND THE BITS THAT FAILED
2756 005724 050013 RIS R0,(R3) ;AND PLACE IT ON THE STACK
2757 005726 012700 001766 MOV R0,=FNDPPG,=24,R0 ;THIS CODE BRINGS THE RELATIVE ADDRESS
2758 005732 000700 000022 68: ADD PC,R0 ;OF THE STARTING OF THE ERROR STACK
2759 005734 062700 ADD #18,R0 ;FOR THE SPECIFIC 4K BANK
2760 005740 005316 DEC (SP)
2761 005742 002374 RGE 68
2762 005744 005726 TST (SP)+ ;RESTORE THE STACK POINTER
2763
2764 005746 105037 000277 ERRTRY: CLR R0 ;DISABLE ANY TYPE OUT
2765 005752 105737 000300 18: TSTR R0,=PRERR ;IF THIS IS PARITY PROBLEM
2766 005756 001007 BNE 38 ;THEN GO TO 38
2767 005760 105720 000301 TSTR R0,=(R0)+ ;OTHERWISE INCREMENT THE ERROR STACK POINTER BY 1
2768 005762 105737 000301 TSTR R0,=BADERR ;IF THIS IS ADDRESSING PROBLEM
2769 005766 001003 BNE 38 ;THEN GO TO 38
2770 005770 105720 TSTR R0,=(R0)+ ;INCREMENT THE POINTER R0 BY 1
2771 005772 005713 28: TST (R3) ;IS BIT 15 OF (R3) SET?
2772 005774 100015 BPL 48 ;IF NOT THEN GO TO 48
2773 005776 122710 000377 38: CMPB #37,(R0) ;OTHERWISE SEE IF THIS ERROR HAS OCCURED 377 TIMES
2774 006002 001401 BEQ 58 ;IF SO DON'T BUMP ERROR COUNT
2775 006004 105210 INCR R0 ;INCREMENT THE ERROR COUNTER BY 1
2776 006006 122710 000001 58: CMPB #1,(R0) ;MORE THAN 1 ERROR OCCURRED ON THIS BIT?
2777 006012 001404 BEQ 78 ;BRANCH IF NO
2778 006014 032777 000400 172426 BIT #400,SWR ;STOP ERROR PRINTOUT AFTER 1 WANTED?
2779 006022 001002 BNE 48 ;BRANCH IF YES (DON'T TYPE ERRPR)
    
```

```
2780 006024 105237 000277 781 INCB #*TYPEMB ;ENABLE THE TYPE OUT ROUTINE
2781 006030 105737 000300 481 TSTB #*SPRERR ;PARITY ERROR?
2782 006034 001403 BEQ 96 ;BRANCH IF NO
2783 006036 004767 000402 JSR PC,TPPRER ;ELSE TYPE "PAR ERR"
2784 006042 000411 BR 88 ;AND DON'T TEST INDIVIDUAL BIT FAILURES.
2785 006044 105737 000301 981 TSTB #*SADERR ;ADDRESS ERROR?
2786 006050 001403 BEQ 68 ;BRANCH IF NO
2787 006052 004767 000342 JSR PC,TPADERR ;PRINT "ADR ERR"
2788 006056 000403 BR 88
2789 006060 105720 681 TSTB (P0)+ ;POINT TO NEXT ENTRY IN ERROR STACK
2790 006062 006313 ASI (R3) ;IS THERE STILL AN ERROR BIT SET IN ERROR.
2791 006064 001342 PNE 78 ;BR IF YFS - KEEP FILLING ERROR STACK
2792 006066 112737 000006 000314 881 MOVB #6,#*TYPCNT ;TELL TYPOCT TO TYPE 6 WORDS OF ERROR STACK.
2793
2794 006074 004767 001142 JSR PC,PUTADP ;THE STACK POINTED BY R3
2795 ;GO TO THE SUBROUTINE TO PLACE THE ADDRESS IN R1
2796 006100 004767 000616 JSR PC,TYPEERR ;AT LOCATIONS (R3) AND (R3-2)
2797 ;TYPE ERROR STACK (7 WORDS)
2798 006104 005037 000300 1081 CLR #*SPRERR ;CLEAR ADDRESS/PARITY ERROR FLAGS
2799 006110 012600 MOV (SP)+,R0 ;RESTORE R0
2800 006112 012603 MOV (SP)+,R3 ;AND R3
2801 006114 105737 000420 FNDERR: TSTB #*EMV ;ARE WE RUNNING UNDER APT?
2802 006120 001404 REQ 28 ;IF NOT THEN TEST FOR HALT
2803 006122 012737 000001 000400 MOV #1,#*MSGTY ;OTHERWISE INFORM THE APT
2804 006130 000442 BR FATHLT ;GOTO FATHLT AND WAIT FOR APT.
2805
2806 006132 010246 281 MOV R2,-(SP) ;SAVE R2 TEMP
2807 006134 005777 172310 TST R0WP ;DOES THE OPERATOR REQUIRE THE PROGRAM TO HALT
2808 ;ON ERROR
2809 006140 100405 RMI 48 ;IF SO THEN HALT ON ERROR
2810 ;CHECK FOR CONTROL-C KEY
2811
2812 006142 004767 001526 JSP PC,CHECKC ;IF CONTROL-C TYPED THEN PRINT ERROR HISTORY
2813 ;AND HALT AT FATHLT.
2814 006146 105737 000442 781 TSTB #*42 ;ARE WE RUNNING UNDER ACT?
2815 006152 001401 BEQ 68 ;BRANCH IF NO
2816
2817 006154 000000 481 HALT ;PROGRAM HAS HALTED ON ERROR, R1 IS POINTING
2818 ;TO A LOCATION WHICH SHOULD HAVE CONTAINED
2819 ;THE WORD STORED IN R0
2820 006156 012602 681 MOV (SP)+,R2 ;RESTORE R2
2821 006160 062716 000002 ADD #2,(SP) ;RESTORE THE RETURN ADDRESS
2822 006164 000207 RTS PC ;RETURN FROM THE SUBROUTINE
2823
2824
2825
2826 006166 FATERR:
2827 006166 004767 000412 SEQERR: JSR PC,TPCRLF ;TYPE "ERR #"
2828 006172 051105 020122 .ASCIZ /FRR #/
2829 .EVEN
2830
2831 006200 017637 000000 000402 MOV R(SP),#*SFATAL ;LOAD THE LOCATION SFATAL WITH THE ERROR NUMBER
2832 006206 105237 000314 INCB #*TYPCNT ;TELL STPNUM TO TYPE 1 WORD
2833 006212 012703 000376 MOV #376,R3 ;STPNUM USES R3 AS STACK
2834 006216 013743 000402 MOV #*SFATAL,-(R3) ;PUT ERROR NO. ON STACK
2835 006222 005743 TST -(R3) ;STPNUM REQUIRES THIS
```

```
2836 006224 004767 000560 JSR PC,FATYP ;TYPE ERROR NO.
2837 006230 105737 000420 APTHLT: TSTB #*ENV ;RUNNING UNDER APT?
2838 006234 001327 BNE FNDERR ;BRANCH IF YES
2839 006236 000000 FATHLT: HALT ;FATAL ERROR OR "C HALT.
2840 006240 000137 000250 JMP #*PESTPT ;RESTART TST BUT DON'T CLEAR PASS COUNT
2841 ;IN CASE "C RESTART.
2842
2843
2844 ;PARERR
2845 ; PARITY TRAP HANDLER
2846 ; COME HERE FROM A TRAP TO 114.
2847 ; THIS ROUTINE SEARCHES THE AVAILABLE PARITY MODULES AND IF ONE
2848 ; HAS A PARITY ERROR BIT SET THE GET THE PARITY ERROR ADDRESS
2849 ; AND CALL THE "ERROR" ROUTINE TO PRINT ERROR MESSAGE.
2850 ; IF NO PARITY ERROR BITS CAN BE FOUND A FATAL ERROR IS DONE.
2851 ;
2852 ; REGISTER US AGE.
2853 ; R0= HOLDS PARITY MODULE ADDRESSES
2854 ; R1= GETS ERROR ADDRESS FOR "ERROR" CALL.
2855
2856 006244 012637 000356 PARERR: MOV (SP)+,#*PARSP ;SET PARSP TO RETURN ADDRESS
2857 006250 011637 000360 MOV (SP),#*PARPS ;SAVE PSW FOR RETURN
2858 006254 013706 000350 MOV #*SAVR6,SP ;AND RESET THE SP SINCE NOT ENOUGH STACK ROOM
2859 ; TO COMPLETE THE ERROR SERVICE ROUTINE.
2860 006260 010067 000130 MCV R0,SAVR0 ;SAVE R0 DURING PARITY SERVICE
2861 006264 010167 000126 MCV R1,SAVR1 ;SAVE R1 DURING PARITY SERVICE
2862 006270 013701 000352 MOV #*PARMAP,R1 ;GET PARITY AVAILABLE MAP
2863 006274 012700 172100 MOV #172100,R0 ;R0= FIRST PARITY ADDRESS.
2864
2865 006300 005701 TST R1 ;ANY PARITY MODULES AVAILABLE?
2866 006302 001441 BEQ 48 ;BR IF NO "FATAL ERROR"
2867 006304 000601 ROR 28 ;SHIFT PARITY MAP BIT INTO C BIT.
2868 006306 103005 BCC 26 ;BRANCH IF THIS PARITY MODULE NOT AVAILABLE.
2869 006310 005710 TST (R0) ;PARITY MODULE ERROR BIT SET?
2870 006312 100406 BMI 38 ;BRANCH IF YES -CALL "ERROR" ROUTINE
2871 006314 020027 172136 CMP R0,#172136 ;DONE ALL PARITY MODULES?
2872 006320 020032 RCE 48 ;BR IF YFS= GO TO FATAL ERROR CALL-
2873 006322 062700 000002 281 ADD #2,R0 ;POINT TO NEXT PARITY ADDRESS
2874 006326 000766 BR 18 ;AND KEEP TRYING
2875 006330 042710 100000 381 RIC #100000,(R0) ;CLEAR PARITY ERROR BIT.
2876 006334 011001 MCV (R0),R1 ;GET PARITY MODULE Csp
2877 006336 006101 POL R1 ;SHIFT ERROR ADDRESS BITS 11-5 INTO 15-9
2878 006340 006101 POL R1
2879 006342 006101 FOL R1
2880 006344 006101 ROL R1
2881 006346 042701 000777 RIC #777,R1 ;SAVE ERROR ADDRESS ONLY
2882 006352 105237 000300 INCB #*SPRERR ;TELL "ERROR" PARITY ERROR CALL.
2883 006356 004767 172246 JSP PC,TRPOP ;*ERROR* REPORT ERROR MESSAGE
2884 006362 000505 SP *****ERROR NUMBER 50*****
2885
2886 006364 016700 000024 MOV SAVR0,R0 ;RESTORE R0
2887 006370 016701 000022 MOV SAVR1,R1 ;RESTORE R1
2888 006374 013746 000360 MOV #*PARPS,(SP) ;SET RETURN PSW ON STACK
2889 006400 013746 000356 MOV #*PARSP,(SP) ;AND SET RETURN ADDRESS ON STACK
2890 006404 000002 RTI ;RETURN TO TEST WHERE PARITY TRAP OCCURRED.
2891
```



```

2892 ;COME HERE IF NO PARITY ERROR FLAG FOUND SET
2893 48:
2894 006406 004767 177554 JSR PC,FATERR ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
2895 006412 000051 51 ;*****ERROR NUMBER 51*****
2896
2897 ;R0+P1 ARE SAVED HERE FOR PARITY TRAP DUE TO INSUFFICIENT
2898 ;STACK SPACE BETWEEN 500-450.
2899 SAVR0: 0 ;SAVE R0 DURING PARITY TRAP SERVICE
2900 006414 000000 SAVR1: 0 ;SAVE R1 DURING PARITY TRAP SERVICE
2901 006416 000000
2902
2903
2904 006420 105737 000277 TPADER: TSTR 0#TYPENR ;TYPE ERROR?
2905 006424 001406 BEQ 18 ;BRANCH IF NO
2906 006426 004767 000160 JSR PC,PNTMES ; TYPE CR, LF AND "ADR ER"
2907 006432 042101 020122 051105 .ASCIZ /ADR ERR/
2908 006440 000122
2909
2910 006442 000207 18: .EVEN
RTS PC
2911
2912 006444 105737 000277 TPPREP: TSTR 0#TYPENB ;ERROR PRINTOUTS ALLOWED?
2913 006450 001406 BEQ 18 ;BRANCH IF NO
2914 006452 004767 000134 JSR PC,PNTMES ;GO TO TYPE CR, LF AND "PAR ERR"
2915 006456 040520 020122 051105 .ASCIZ /PAR ERR/
2916 006464 000122
2917
2918 006466 000207 18: .EVEN
RTS PC
  
```

```

2919
2920 ;* TYPE OUT ROUTINE
2921 ;* -----
2922 ;*
2923 ;* THIS ROUTINE IS USED BY THE PROGRAM TO TYPE OUT ANY CHARACTER
2924 ;*
2925 ;*
2926
2927 006470 010146 NOTYP: MOV R1,-(SP)
2928 006472 016001 000002 MOV R0,(SP),R1
2929 006476 105771 48: TSTR (R1)+ ;IF THIS TYPE OUT HAS BEEN SUPPRESSED THEN
006500 001376 BNE 48 ;PREPARE TO RETURN
2931 006502 000412 BR RETTYP
2932 006504 010146 8TYPE: MOV R1,-(SP) ;SAVE R1
006506 010046 MOV R0,-(SP) ;AND R0 ON THE STACK
2934 006510 016001 000004 MOV R0,(SP),R1 ;PLACE THE ADDRESS OF MESSAGE TO BE TYPED IN R1
2935 006514 112100 25: MOVE (R1)+,R0 ;PLACE THE BYTE TO BE TYPED IN R0
2936 006516 001403 REQ 48 ;IF IT IS END OF MESSAGE THEN GO TO 48
006520 004767 000022 JSR PC,STPCHR ;OTHERWISE GO TO TYPE THE CONTENTS OF R0
2938 006524 000773 BR 28
2939 006526 012600 48: MOV (R0)+,R0 ;PESTORE R0
2940 006530 005201 RETTYP: INC R1 ;CAUSE R1 TO
006532 042701 000001 BIC #1,R1 ;POINT TO EVEN ADDRESS
2942 006536 010166 000002 MOV R1,2(SP) ;MODIFY THE RETURN ADDRESS
2943 006542 012601 MOV (SP)+,R1 ;PESTORE R1
2944 006544 000416 BR EXTYP ;AND RETURN VIA RTS PC
2945
2946 006546 132737 000040 000421 8TPCHR: BTR #40,0#ENVM ;HAVE TYPE OUTS BEEN DISABLED?
2947 006554 001005 48 RNE 48 ;IF SO THEN RETURN FROM THE SUBROUTINE
2948 006556 105737 177564 28: TSTR #8#TPS ;WAIT HERE
2949 006562 100375 28 BPL 28 ;UNTIL THE PRINTER IS READY
2950 006564 110037 177566 R0,0#STPB ;LOAD DATA TO BE TYPED INTO DATA REG.
2951 006570 000004 48: BR EXTYP ;RETURN
2952
2953 006572 004767 177706 PCRLF: JSR PC,8TYPE
2954 006576 005015 000 .ASCIZ <15><12> ;CR/LF
2955 006602 000207 .EVEN
2956 EXTYP: RTS PC ;RETURN
2957
2958 006604 004767 177762 TPCLRF: JSR PC,PCRLF ;TYPE CR/LF
2959 006610 000735 BR 8TYPE ;NOW GO TO TYPE THE REST OF THE MESSAGE
2960
2961
2962 006612 032777 000020 171630 PNTMES: BIT #20,#SWR ;PRINTOUTS ALLOWED?
2963 006620 001323 BNE NOTYP ;BRANCH IF NO
2964 006622 123737 000042 000046 CMPR #42,#46 ;RUNNING UNDER ACT 11?
2965 006630 001717 REQ NOTYP ;BRANCH IF YES =NOT PRINTOUT=
2966 006632 000764 BR TPCLRF ;SEND CR/LF AND TYPE MESSAGE.
  
```

```

2967
2968
2969
2970
2971
2972
2973
2974
2975 006634 004767 177732 TYPDEC: JSR PC,PCRLF ;TYPE CR/LF
2976
2977 006640 005046 6TPDEC: CLR -(SP)
2978 006642 013746 000312 MOV #DECWRD, -(SP) ;GET THE WORD THAT HAS TO BE CONVERTED TO A
;DECIMAL NUMBER
2979
2980 006646 162716 000012 28: SUB #10, (SP)
2981 006652 002403 BLT 48 ;IF THE NUMBER IN (SP) WAS LESS THAN 10, THEN
;GO TO 48
2982
2983 006654 005266 000002 INC 2(SP) ;OTHERWISE ADD 1 TO THE LOCATION STORING 10'S DIGIT
2984 006660 000772 RR 28 ;AND RETURN TO 28
2985 006662 062716 000012 48: ADD #10, (SP)
2986 006666 052716 000060 BIS #60, (SP) ;MAKE THE CONTENTS OF (SP) A DECIMAL NUMBER
2987 006672 112667 000020 MOVB (SP)+68-2 ;PLACE THE 1'S DIGIT TO BE TYPED
2988 006676 052716 000060 BIS #60, (SP) ;MAKE THE CONTENTS OF (SP) A DECIMAL NUMBER
2989 006702 112667 000007 MOVB (SP)+68-3 ;PLACE THE 10'S DIGIT TO BE TYPED
2990 006706 004767 177572 JSR PC,8TYPE ;GO TO TYPE THE NUMBER IN DECIMAL FOLLOWED BY
;3 SPACES
2991
2992 006712 020040 030040 000060 .ASCIZ / 00/
2993 .EVEN
2994 006720 000207 68: RTS PC ;RETURN FROM THE SUBROUTINE
  
```

```

2995
2996
2997
2998
2999
3000
3001
3002
3003
3004
3005
3006
3007
3008
3009
3010 006722 032777 020000 171520 TYPEPPI BIT #20000,05WR ;FRPOP PRINTOUT WANTED?
3011 006730 001054 PNE OCTXT ;BRANCH IF NO
3012 006732 004767 177634 JSP PC,PCRLF ;TYPE CR/LF
3013 006736 004767 000012 JSR PC,TYPEOCT ;TYPE OCTAL NO.
3014 006742 000447 BF OCTXT ;RETURN VIA RTS PC
3015 006744 012123 OCTYPI: MOV (R1)+, (R3)+ ;PLACE THE HIGH ORDER BITS AT LOCATION POINTED
;BY R3
3016
3017 006746 012113 MOV (R1)+, (R3) ;AND NOW PLACE THE LOW ORDER BITS
3018 006750 105237 000314 TPCNT INCB #TYPECNT ;ENABLE THE TYPE OUT OF ONE OCTAL WORD
3019 006754 052743 000004 TYPOCT: BIS #4, -(R3)
3020 006760 106113 28: ROLR (R3)
3021 006762 103376 RCC 28
3022 006764 005000 CLR R0
3023 006766 106113 ROLB (R3) ;GET BITS 17 & 16 INTO R0
3024 006770 006100 ROL R0
3025 006772 106113 ROLB (R3)
3026 006774 006100 ROL R0
3027 006776 000405 177500 RPTOCT: JSR #8TPNUM
3028 007000 004767 177500 JSR PC,8TYPE ;TYPE 3 SPACES
3029 007004 020040 000040 .ASCIZ /
;EVEN
3030
3031 007010 005000 FATYPI: CLR R0
3032 007012 012723 000006 6TPNUM: MOV #6, (R3)+ ;ENABLE THE TYPE OUT OF 6 OCTAL DIGITS
3033 007016 000241 48: CLC
3034 007020 006113 ROL (R3)
3035 007022 006100 ROL R0 ;PLACE THE CARRY FROM (R3) IN R0
3036 007024 052700 000060 BIS #60, R0 ;FOR THE CONTENTS OF R0 WITH AN ASCII 0
3037 007030 004767 177512 JSR PC,8TPCHR ;TYPE THE OCTAL NUMBER STORED IN R0
3038 CLR R0
3039 007036 006113 ROL (R3)
3040 007040 006100 ROL R0 ;PLACE THE CARRY FROM (R3) IN R0
3041 007042 006113 ROL (R3)
3042 007044 006100 ROL R0
3043 007046 105363 177776 DECB #7(R3) ;PLACE THE CARRY FROM (R3) IN R0
3044 007052 001361 BNE 48 ;IF WE HAVEN'T TYPED THE 6 OCTAL DIGITS
3045 007054 105337 000314 DECB #TYPECNT ;IF ALL THE WORDS REQUIRED HAVE NOT BEEN
;TYPED THEN REPEAT FROM RPTOCT
3046 007060 001347 RPTOCT: BNE
3047 007062 000207 OCTXT: RTS PC
  
```

```

3048
3049
3050
3051
3052
3053
3054
3055
3056
3057 007064 012702 001400 MEMMNG: MOV #1400,R2
3058 007070 105037 000276 MMRREG: CLR #MMVAVA ;CLEAR THE BYTE THAT IS SUPPOSED TO INDICATE
; THAT MEM. MANAG. IS AVAILABLE FOR TESTING
; HAS THE OPERA. ASKED TO CHECK MEMORY MANAG. ?
; IF NOT THEN RETURN FROM THE SUBROUTINE
; PREPARE TO SETUP TIME OUT VECTOR
; RETURN ADDRESS TO NOMM
; AND WITH A PSW OF 340
; TRY TO REACH MEM. MANAG. SR0
; IF IT IS AVAILABLE THEN SET MEM. MANAG. AVAILABLE
; BYTE
; R1 IS POINTING TO PAR0
; PAR0 WILL POINT TO BANK 0
3059 007074 032777 010000 171346 BIT #10000,PSWR
3060 007102 001441 REQ RETMM
3061 007104 012700 000004 MOV #4,R0
3062 007110 012720 007210 MOV #NOMM,(R0)+
3063 007114 012710 000340 MOV #340,(R0)
3064 007120 005037 177572 CLR #SR0
3065 007124 105237 000276 INCB #MMVAVA
3066
3067
3068 007130 012701 172340 MOV #172340,R1
3069 007134 005021 CLR (R1)+
3070 007136 062702 000200 201 ADD #200,R2
3071 007142 010221 MOV P2,(R1)+
3072 007144 020127 172356 CMP R1,#172356
3073 007150 103772 RLO 26
3074 007152 012711 007600 MOV #7600,(R1)
3075 007156 012701 172300 MOV #172300,R1
3076 007162 012721 077406 401 MOV #77406,(R1)+
3077 007166 020127 172316 CMP R1,#172316
3078 007172 101773 BLOS 48
3079 007174 005237 177572 INC #SR0
3080 007200 005010 SRETMM: CLR (R0)
3081 007202 012740 000104 MOV #USER,-(R0)
3082 007206 000207 RETMM: RTS PC
3083
3084 007210 022626 NOMM: CMP (SP)+,(SP)+
3085 007212 004767 177366 JSP PC,TPCLR
3086 007216 047516 045440 000124 .ASCIZ /NO KT/
3087 .EVEN
3088 007224 004767 176736 JSR PC,FATERR
3089 007230 000052 52 ;*****ERROR NUMBER 52*****
3090
3091 007232 000762 BR SRETMM
3092
3093 007234 013702 172354 UPMN: MOV #172354,R2
3094 007240 000713 BR MMRREG
    
```

```

3095
3096
3097
3098
3099
3100
3101
3102
3103
3104
3105
3106 007242 005063 177776 PUTADR: CLR =(R3)
3107 007246 010113 MOV R1,(R3)
3108 007250 105737 000276 TSTB #MMVAVA
3109 007254 001425 REQ 68
3110 007256 010146 MOV R1,-(SP)
3111 007260 042701 017777 BIC #17777,R1
3112 007264 040113 BIC R1,(R3)
3113 007266 052701 004000 RLS #1000,R1
3114 007272 006001 POR R1
3115 007274 103376 BCC 28
3116 007276 062701 172340 ADD #172340,R1
3117 007302 011101 MOV (R1),R1
3118 007304 052701 010000 BIS #10000,R1
3119 007310 006101 ROL R1
3120 007312 103376 BCC 48
3121 007314 006101 ROL R1
3122 007316 006143 ROL =(R3)
3123 007320 006101 ROL R1
3124 007322 006123 ROL (R3)+
3125 007324 050113 BIS R1,(R3)
3126 007326 012601 MOV (SP)+,R1
3127 007330 000207 601 RTS PC
3128
3129
3130
3131
3132
3133
3134
3135
3136
3137
3138
3139
3140
3141 007332 016143 000001 GETADR: MOV 1(R1),-(R3)
3142 007336 005043 CLR -(R3)
3143
3144 007340 116113 177777 MOVB #1(R1),(R3)
3145 007344 000207 201 RTS PC
    
```

```

3146
3147 ;* CONVERT 18 BIT ADDRESS TO THE PAR FORM
3148 ;*
3149 ;*
3150 ;* THIS SUBROUTINE IS USED TO CONVERT 18 BIT ADDRESS STORED IN
3151 ;* LOCATIONS POINTED BY R3 AND R3+2 TO THE FORM IT WILL BE STORED
3152 ;* IN A PAR. THE RESULT IS LEFT IN R2, R1 IS LOADED WITH BITS
3153 ;* 0-12 OF THE ADDRESS AND R0 WITH 160000
3154 ;*
3155
3156 007346 005046 SGT SIZE CLR -(SP) ;PREPARE TO PLACE ADDRESS BITS 13-17 IN BITS
3157 ;0-4 OF R2
3158
3159 007350 012301 GETSIZ: MOV (P3)+,R1
3160 007352 011302 MOV (L3)+,R2 ;LOAD R2 WITH THE LOW ORDER BITS OF THE ADDRESS
3161 007354 042702 017777 RIC #17777,R2 ;CLFAR ADDRESS BITS 0-12
3162 007360 052702 000040 28: RLS #40,R2
3163 007364 006001 48: ROR R1
3164 007366 006002 ROR R2 ;ROTATE R1 AND R2 7 TIMES
3165 007370 103375 BCC 48
3166 007372 005716 TST (SP) ;IF RETURN PC IS ZERO THEN IT MUST BE THE
3167 007374 001004 RNE 68 ;FLAG THAT WAS SET AT SGT SIZE
3168 007376 005726 TST (SP)+ ;POD THE FLAG OFF STACK
3169 007400 052702 000100 RLS #100,P2 ;KEEP ROTATING
3170 007404 000767 BR 48
3171 007406 012301 68: MOV (P3)+,R1 ;PLACE THE LOW ORDER ADDRESS BITS IN R1
3172 007410 012700 160000 MOV #160000,R0
3173 007414 040001 RIC R0,R1 ;LEAVE BITS 0-12 OF THE ADDRESS IN R1
3174 007416 000207 RTS PC ;RETURN FROM THE SUBROUTINE
3175
3176
3177 ;* SUBROUTINE TO DISABLE MEMORY MANAGEMENT
3178 ;*
3179 ;*
3180 ;* THIS SUBROUTINE IS CALLED TO DISABLE THE MEMORY MANAGEMENT
3181 ;* UNIT
3182 ;*
3183
3184 007420 105737 000276 CLRMM: TSTB 0#MMAVA ;WAS THE MEMORY MANAGEMENT ENABLED ?
3185 007424 001404 REQ 16 ;IF NOT THEN GO TO 15
3186 007426 005037 177572 CLR 0#SR0 ;DISABLE THE MEMORY MANAGEMENT
3187 007432 105037 000276 CLRMM CLR 0#MMAVA ;AND DO NOT ATTEMPT TO TEST MEM. MANAG.
3188 007436 000207 15: RTS PC ;RETURN FROM THE SUBROUTINE
3189
3190
3191 ;* GET BANK NO. UNDER TEST
3192 ; CALLED BY PRTYP AND TST13 TO GET BANK NO. UNDER TEST INTO PBNK.
3193 ;REGISTERS
3194 ;R0=POINT TO PAR UNDER TEST
3195 ;R3=VIRTUAL ADDRESS ON ENTRY
3196 ;R0+R3 ARE RESTORED ON EXIT.
3197
3198 007440 010046 GETBNK: MOV R0,-(SP) ;SAVE R0
3199 007442 010346 MOV R3,-(SP) ;SAVE R3
3200 007444 042703 017777 RIC #17777,P3 ;SAVE ONLY VIRTUAL BANK BITS
3201 007450 052703 010000 RLS #10000,P3 ;SETUP R3 SHIFT BIT

```

```

3202 007454 000241 CLC R3 ;SHIFT A BANK BIT
3203 007456 006003 15: ROR R3 ;UNTIL IN BITS <210> OF P3
3204 007460 103376 RCC 16 ;MEMORY MANAGEMENT UNDER TEST?
3205 007462 105737 000276 TSTB 0#MMAVA ;NO EXIT
3206 007466 001407 REQ 25
3207
3208 ;GET PAR ADDRESS AND PHYSICAL BANK NO.
3209 007470 006303 ASL R3 ;MAKE R3 PAR ADDRESS OFFSET.
3210 007472 062703 172340 ADD #172340,P3 ;MAKE FULL PAR ADDRESS.
3211 007476 011300 MOV (P3),R0 ;GET PAR CONTENTS
3212 007500 006300 ASL R0
3213 007502 000300 SWAB R0 ;SHIFT BANK BITS TO BITS <710>
3214 007504 110003 MOV R0,R3 ;SET R3 TO PHYSICAL BANK NO.
3215 007506 010337 000312 28: MOV R3,0#PBK ;STORE PHYSICAL BANK NO.
3216 007512 012603 MOV (SP)+,R3 ;RESTORE R3
3217 007514 012600 MOV (SP)+,R0 ;RESTORE R0
3218 007516 000207 RTS PC ;RETURN TO CALLER
3219
3220
3221 ; PARITY ENABLE/DISABLE ROUTINE
3222 ;
3223 ; THIS ROUTINE ENABLER OR DISABLES PARITY MODULES AND PRINTS ASSOCIATED MESSAGES,
3224 ; IF PARITY AVAILABLE THEN BIT13 OF "REL" IS SET AND "PAR"ITY IS PRINTED.
3225 ; ALSO THE BACKGROUND TEST PATTERN (LOC. PARPAT) IS SET=376
3226 ;
3227 ;
3228 ; REGISTER USAGE.
3229 ; R0= POINTS TO BUS TIMEOUT TRAP VECTOR (LOC. 4)
3230 ; R1= HOLDS PARITY MODULE UNIBUS ADDRESS.
3231 ; R2= ON ENTRY HOLDS ENABLE/DISABLE CODE.
3232 ; IF R2=0 THEN DISABLE
3233 ; IF R2=1 THEN ENABLE
3234 ; R3= SCRATCH TO SETUP LOC. PARMAP WITH A MAP OF PARITY MODULES PRESENT.
3235
3236 ;CALL IS
3237 ; MOV #1,R2 ;ENABLE CODE
3238 ; JSP PC,PARITY
3239
3240
3241 007520 032777 004000 170722 PARITY: BIT #4000,0SWR ;PARITY TEST WANTED?
3242 007526 001456 BEQ 68 ;BRANCH IF NO
3243
3244 007530 012700 000004 MOV #4,R0 ;POINT R0 TO BUS TIMEOUT ADDRESS.
3245 007534 012710 000116 MOV #58,-6,(R0) ;SET RETURN FROM TIMEOUT TRAP TO 58
3246 007540 006710 000000 ADD PC,(R0) ;IN THE CURRENT BANK,
3247 007542 005037 000352 15: CLR 0#PARMAP ;CLEAR PARITY MAP HOLDER.
3248 007546 012701 172140 MOV #172140,R1 ;SET R1 TO LAST PARITY MODULE ADDRESS+2
3249 007552 012703 100000 MOV #100000,R3 ;SET R3 TO PARMAP AVAILABLE CODE BEGIN.
3250 007556 010241 28: MOV R2,(R1) ;ENABLE A PARITY MODULE+TRAP IF NOT AVAILABLE.
3251 007560 050337 000352 RLS R3,0#PARMAP ;NO TRAP TO 58, SO SET PARITY AVAILABLE.
3252 007564 000241 CLC
3253 007566 006003 38: ROR R3 ;SETUP NEXT PARMAP BIT
3254 007570 103372 RCC 28 ;BRANCH IF NOT DONE ALL PARITY ADDRESSES.
3255 007572 012710 000104 MOV #RUSER,(R0) ;RESET BUS TIMEOUT TRAP VECTOR
3256 007576 005702 TST R2 ;IS THIS A DISABLE CALL?
3257 007600 001431 BEQ 68 ;BRANCH IF YES (EXIT)

```

```

3258 007602 005737 000352          TST  @PARMAP      ;WERE ANY PARITY MODULES FOUND?
3259 007606 001011                    RNE  48           ;BRANCH IF YES
3260 007610 004767 176770          JSR  PC,TPCRLF   ;PRINT "NO PAR"
3261 007614 047516 050040 051101  .ASCIZ /NO PAR/
3262 007622 000000
3263 007624 007624                    .EVEN
3264 007624 004767 176336          JSP  PC,FATERR   ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
3265 007630 000053                    53           ;*****ERROR NUMBER 53*****
3266
3267
3268 007632 152737 000040 000405 48:  BISR  #40,0#REL   ;SET PARITY UNDER TEST FLAG
3269 007640 012737 000376 000316  MOV  #376,0#RAKPAT ;SET BACKGROUND PATTERN TO
3270                                ;WORST CASE PARITY CODE.
3271 007646 004767 176732          JSR  PC,TPCRLF   ;PRINT "TST PARITY"
3272 007652 040520 000122          .ASCIZ /PAR/
3273                                .EVEN
3274 007656 000405                    BR  EXITC        ;AND EXIT VIA RTS PC
3275
3276                                ;GET HERE IF PAPIY ADDRESS TIMED OUT TO LOC. 4
3277

```

```

3278 007660 022626                    58:  CMP  (SP)+,(SP)+ ;RESET STACK FROM TRAP
3279 007662 000741                    BR  38           ;KEEP TRYING PARITY ADDRESSES.
3280
3281 007664 142737 000040 000405 68:  BICR  #40,0#REL   ;CLEAR PARITY TESTING FLAG
3282 007672                                EXITC:
3283 007672 000207                    78:  RTS  PC        ;RETURN TO CALLER
3284
3285
3286
3287
3288                                ;CHECKC
3289                                ; THIS ROUTINE CHECKS IF CONTROL-C WAS TYPED AT THE END OF EACH
3290                                ; TEST OP IN THE ERROR TYPE ROUTINE.
3291                                ; IF CONTROL-C TYPED THE PROGRAM IS RETURNED TO LOWER MEMORY IF IT WAS
3292                                ; RELOCATED AND THE ERROR HISTORY IS PRINTED OUT.
3293                                ; FINALLY IT HALTS AT FATHLT.
3294
3295 007674 105037 000315          CHECKC: CLR  @SAVKBR   ;INIT CONTROL-C FLAG.
3296 007700 105737 177560          TSTB @TKS        ;ANY CHAR. TYPED?
3297 007704 100372                    BPL  EXITC        ;BR IF NO-EXIT VIA RTS PC-
3298 007706 113702 177562          MOV  @#KBR,R2    ;GET THE CHAR TYPED.
3299 007712 042702 000200          BIC  #200,R2     ;CLEAR THE PARITY BIT.
3300 007716 122702 000003          CMP  #3,R2      ;IS IT CONTROL-C?
3301 007722 001363                    BNE  EXITC        ;BRANCH IF NO -EXIT VIA RTS PC-
3302 007724 110237 000315          MOV  R2,@SAVKBR ;ELSE STORE THE CHAR. FOR USE AS A FLAG.
3303 007730 004767 176650          JSR  PC,TPCRLF   ;PRINT "C"
3304 007734 041536 000000          .ASCIZ /C/
3305 007740 000740
3306 007740 000167 175132          JMP  RELOER      ;GO RETURN PROGRAM TO LOWER CORE IF RELOCATED.
3307
3308                                ;=7744
3309 007744 000000          ENDPRG:  ;THIS BEGINS THE STORAGE FOR THE ERROR HISTORY
3310                                ;STACK, FOR EACH 4K BANK 10. BYTES ARE SAVED.
3311                                ;ALSO THE ABSOLUTE LOADER AND XSDP CODE IS SAVED
3312                                ;AFTER THE ERROR STACK.
3313                                ;FOR 4K MEMORY SIZE THEN PROGRAM=7744+22=7776
3314 000001                                .END

```


| | | | | | | | | | | | | | | | | | | |
|--------|--------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|------|-------|--|--|--|--|
| NOMM | 007210 | 2476# | 2883 | 2886# | 2893 | 2897# | 3088 | 3091# | 3264 | 3267# | | | | | | | | |
| NOTYP | 006470 | 2927# | 2963 | 2965 | | | | | | | | | | | | | | |
| OCTYP | 006744 | 1387 | 1391 | 3015# | | | | | | | | | | | | | | |
| OCTXT | 007062 | 3011 | 3014 | 3047# | | | | | | | | | | | | | | |
| ONEPAS | 000556 | 1260 | 1263# | | | | | | | | | | | | | | | |
| PARERR | 006244 | 1256 | 2856# | | | | | | | | | | | | | | | |
| PARFL | 005470 | 2656 | 2675# | | | | | | | | | | | | | | | |
| PARITY | 007520 | 1490 | 2609 | 3241# | | | | | | | | | | | | | | |
| PARMAP | 000352 | 1105# | 2862 | 3247# | 3251* | 3258 | | | | | | | | | | | | |
| PARPS | 000360 | 1188# | 2857# | 2888 | | | | | | | | | | | | | | |
| PARSP | 000356 | 1187# | 2856# | 2889 | | | | | | | | | | | | | | |
| PASFLG | 000306 | 1138# | 1142 | 1502# | 1860# | 1928# | 1945# | 1977# | 1988 | 1990# | 2003# | 2024# | 2047 | 2063# | | | | |
| | | 2065 | 2070 | 2285# | 2302 | 2322 | 2328# | 2329 | 2337# | 2338 | 2427 | 2434 | 2449 | 2467 | | | | |
| | | 2484 | 2494# | 2505# | 2522# | 2746 | | | | | | | | | | | | |
| PBNK | 000312 | 1145# | 2739 | 3215# | | | | | | | | | | | | | | |
| PCRLF | 006572 | 1306 | 2953# | 2958 | 2975 | 3012 | | | | | | | | | | | | |
| PNTMES | 006612 | 1034 | 2246 | 2512 | 2542 | 2906 | 2914 | 2962# | | | | | | | | | | |
| PJTADR | 007242 | 1374 | 1377 | 2794 | 3106# | | | | | | | | | | | | | |
| PWRDN | 000070 | 1000# | 1022 | | | | | | | | | | | | | | | |
| PWRUP | 000136 | 1000 | 1033# | | | | | | | | | | | | | | | |
| REL | 000405 | 1113# | 1259 | 1465 | 2407 | 2535 | 2537# | 2556 | 2563# | 3268# | 3281# | | | | | | | |
| RELBOT | 000322 | 1163# | 1467 | | | | | | | | | | | | | | | |
| RELOC | 005004 | 1527 | 2529# | | | | | | | | | | | | | | | |
| RELOEP | 005076 | 2536 | 2555# | 3306 | | | | | | | | | | | | | | |
| RESIRT | 000250 | 1048 | 1064 | 1066 | 1072# | 2709 | 2840 | | | | | | | | | | | |
| RESMM | 007206 | 3061 | 3082# | | | | | | | | | | | | | | | |
| RETSTK | 005374 | 2647# | 2651 | 2674 | | | | | | | | | | | | | | |
| RETTY | 006530 | 2931 | 2940# | | | | | | | | | | | | | | | |
| RLODER | 005576 | 2702 | 2711# | 2720 | | | | | | | | | | | | | | |
| RPPOCT | 007000 | 2671 | 3028# | 3046 | | | | | | | | | | | | | | |
| RP10 | 003522 | 2117# | 2203 | 2249 | | | | | | | | | | | | | | |
| RP11 | 004070 | 2199 | 2202 | 2245 | 2248# | | | | | | | | | | | | | |
| RP76 | 003140 | 1975# | 2005 | | | | | | | | | | | | | | | |
| SAVKRB | 000315 | 1150# | 2571 | 2600 | 3295# | 3302# | | | | | | | | | | | | |
| SAVLDR | 001314 | 1410 | 1412 | 1417# | | | | | | | | | | | | | | |
| SAVLOC | 000354 | 1186# | 1998# | | | | | | | | | | | | | | | |
| SAVMAX | 000342 | 1179# | 1477 | | | | | | | | | | | | | | | |
| SAVR0 | 006414 | 2860# | 2886# | 2900# | | | | | | | | | | | | | | |
| SAVR1 | 006416 | 2861# | 2887 | 2901# | | | | | | | | | | | | | | |
| SAVR4 | 000344 | 1180# | 2712 | | | | | | | | | | | | | | | |
| SAVR5 | 000346 | 1073 | 1181# | 1438# | 2555 | 2564# | | | | | | | | | | | | |
| SAVR6 | 000350 | 1033 | 1058 | 1184# | 1251# | 1608 | 2566# | 2858 | | | | | | | | | | |
| SCOPE | 000240 | 983# | | | | | | | | | | | | | | | | |
| SEGERR | 006166 | 1575 | 1659 | 1702 | 1746 | 1834 | 1883 | 1972 | 2021 | 2113 | 2239 | 2279 | 2380 | 2827# | | | | |
| SETSTK | 001220 | 1265 | 1392# | | | | | | | | | | | | | | | |
| SETSWR | 000656 | 1290# | | | | | | | | | | | | | | | | |
| SLFSIZ | 001024 | 1341# | | | | | | | | | | | | | | | | |
| SR0 | 177572 | 1159# | 3065# | 3079# | 3186# | | | | | | | | | | | | | |
| START | 000200 | 1041 | 1058# | | | | | | | | | | | | | | | |
| STRTDI | 000302 | 1134# | 1135 | 2035# | 2064 | | | | | | | | | | | | | |
| SWAPAT | 000320 | 1162# | 2157 | 2161 | 2190# | | | | | | | | | | | | | |
| SWHALT | 001712 | 1555# | | | | | | | | | | | | | | | | |
| SWR | 000450 | 1745# | 1293 | 1296# | 1301# | 1334 | 1448 | 1494 | 1553 | 1557 | 1618 | 2244 | 2248 | 2508 | | | | |
| SWREG | 000176 | 2533 | 2636 | 2693 | 2778 | 2807 | 2962 | 3010 | 3060 | 3241 | | | | | | | | |
| | | 1051# | 1296 | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | | | |
|--------|--------|-------|-------|-------|-------|-------|-------|-------|------|------|------|------|------|------|--|--|--|--|
| SW11 | 004000 | 982# | | | | | | | | | | | | | | | | |
| TBL | 001626 | 1511 | 1515# | 1516 | 1517 | 1518 | 1519 | 1520 | 1521 | 1522 | 1523 | 1524 | 1525 | 1526 | | | | |
| | | 1527 | | | | | | | | | | | | | | | | |
| TKS | 177560 | 1155# | 3296 | | | | | | | | | | | | | | | |
| TPADER | 006420 | 2658 | 2787 | 2904# | | | | | | | | | | | | | | |
| TPCRLF | 006604 | 1067 | 2695 | 2827 | 2958# | 2966 | 3085 | 3260 | 3271 | 3303 | | | | | | | | |
| TPPRER | 006444 | 2675 | 2783 | 2912# | | | | | | | | | | | | | | |
| TRYSR | 001014 | 1318 | 1334# | | | | | | | | | | | | | | | |
| TSTGO | 001714 | 1554 | 1557# | | | | | | | | | | | | | | | |
| TSTMM | 005156 | 1453 | 2573# | | | | | | | | | | | | | | | |
| TSTREL | 001372 | 1261 | 1438# | | | | | | | | | | | | | | | |
| TSTRP | 000570 | 1264 | 1766# | | | | | | | | | | | | | | | |
| TSTSCP | 001660 | 1545# | 1647 | | | | | | | | | | | | | | | |
| TSTSIZ | 001376 | 1439# | | | | | | | | | | | | | | | | |
| TST0 | 001730 | 1515 | 1573# | | | | | | | | | | | | | | | |
| TST1 | 002166 | 1516 | 1655# | | | | | | | | | | | | | | | |
| TST10 | 003502 | 1523 | 2110# | | | | | | | | | | | | | | | |
| TST11 | 004032 | 1524 | 2236# | | | | | | | | | | | | | | | |
| TST12 | 004104 | 1525 | 2277# | 2346 | | | | | | | | | | | | | | |
| TST13 | 004356 | 1526 | 2378# | | | | | | | | | | | | | | | |
| TST2 | 002266 | 1517 | 1699# | | | | | | | | | | | | | | | |
| TST3 | 002376 | 1518 | 1744# | 1823 | | | | | | | | | | | | | | |
| TST4 | 002644 | 1519 | 1832# | | | | | | | | | | | | | | | |
| TST5 | 002754 | 1520 | 1880# | | | | | | | | | | | | | | | |
| TST6 | 003122 | 1521 | 1968# | | | | | | | | | | | | | | | |
| TST7 | 003256 | 1522 | 2018# | | | | | | | | | | | | | | | |
| TYPCNT | 000314 | 1148# | 1254# | 2670# | 2792# | 2832# | 3018# | 3045# | | | | | | | | | | |
| TYPDEC | 006634 | 2662 | 2975# | | | | | | | | | | | | | | | |
| TYPENB | 000277 | 1122# | 1125 | 2764# | 2780# | 2904 | 2912 | | | | | | | | | | | |
| TYPEOP | 005526 | 2695# | | | | | | | | | | | | | | | | |
| TYPERR | 006722 | 2796 | 3010# | | | | | | | | | | | | | | | |
| TYPMEM | 001206 | 1388# | | | | | | | | | | | | | | | | |
| TYPOCT | 006754 | 3013 | 3019# | | | | | | | | | | | | | | | |
| TYPSIZ | 001164 | 1337 | 1335 | 1302# | | | | | | | | | | | | | | |
| TYPSTK | 005354 | 2642# | 2648 | | | | | | | | | | | | | | | |
| UPMM | 007234 | 2578 | 3093# | | | | | | | | | | | | | | | |
| WRTMEM | 000120 | 1025# | 1750 | 1886 | 1975 | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | | | | | |
|---------|--------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|--|--|--|--|--|--|
| \$FATAL | 000402 | 1203* | 2731* | 2740* | 2748 | 2831* | 2834 | | | | | | | | | | | | | |
| \$GTSIZ | 007346 | 1393 | 3156* | | | | | | | | | | | | | | | | | |
| \$HD | 000002 | 973 | | | | | | | | | | | | | | | | | | |
| \$HIBTS | 000276 | 1105* | | | | | | | | | | | | | | | | | | |
| \$HIMAX | 000334 | 1174* | 1315* | | | | | | | | | | | | | | | | | |
| \$KOB | 177562 | 1156* | 3298 | | | | | | | | | | | | | | | | | |
| \$MADR1 | 000432 | 1228* | | | | | | | | | | | | | | | | | | |
| \$MADR2 | 000436 | 1232* | | | | | | | | | | | | | | | | | | |
| \$MADR3 | 000442 | 1235* | | | | | | | | | | | | | | | | | | |
| \$MADR4 | 000446 | 1238* | | | | | | | | | | | | | | | | | | |
| \$MAIL | 000400 | 1061 | 1106 | 1110 | 1201* | 1271 | 1286 | | | | | | | | | | | | | |
| \$MAMS1 | 000430 | 1222* | | | | | | | | | | | | | | | | | | |
| \$MAMS2 | 000434 | 1230* | | | | | | | | | | | | | | | | | | |
| \$MAMS3 | 000440 | 1233* | | | | | | | | | | | | | | | | | | |
| \$MAMS4 | 000444 | 1236* | | | | | | | | | | | | | | | | | | |
| \$MAX | 000336 | 1175* | 1316* | 1401 | | | | | | | | | | | | | | | | |
| \$MBADR | 000300 | 1106* | | | | | | | | | | | | | | | | | | |
| \$MSGAD | 000414 | 1208* | | | | | | | | | | | | | | | | | | |
| \$MSGLG | 000416 | 1209* | | | | | | | | | | | | | | | | | | |
| \$MSGTY | 000400 | 1207* | 1202* | 2803* | | | | | | | | | | | | | | | | |
| \$MTYP1 | 000431 | 1223* | 1324 | | | | | | | | | | | | | | | | | |
| \$MTYP2 | 000435 | 1231* | | | | | | | | | | | | | | | | | | |
| \$MTYP3 | 000441 | 1234* | | | | | | | | | | | | | | | | | | |
| \$MTYP4 | 000445 | 1237* | 1320 | | | | | | | | | | | | | | | | | |
| \$NWTST | 000001 | 1561* | 1563 | 1650* | 1652 | 1691* | 1693 | 1733* | 1735 | 1826* | 1828 | 1866* | 1868 | 1952* | | | | | | |
| | | 1954 | 2007* | 2009 | 2083* | 2085 | 2208* | 2210 | 2255* | 2257 | 2348* | 2350 | | | | | | | | |
| \$PASS | 000406 | 1205* | 1263 | 2692* | 2698 | | | | | | | | | | | | | | | |
| \$PASN | 000304 | 1108* | | | | | | | | | | | | | | | | | | |
| \$PRERR | 000300 | 1126* | 1129 | 1253* | 2765 | 2781 | 2798* | 2882* | | | | | | | | | | | | |
| \$RETH | 007200 | 3084* | 3091 | | | | | | | | | | | | | | | | | |
| \$SVPC | 000044 | 992* | 997 | | | | | | | | | | | | | | | | | |
| \$SWR | 000000 | 973* | 982* | 1574 | 1656 | 1700 | 1745 | 1833 | 1881 | 1969 | 2019 | 2111 | 2237 | 2278 | | | | | | |
| | | 2379 | | | | | | | | | | | | | | | | | | |
| \$SWREG | 000422 | 1213* | 1301 | | | | | | | | | | | | | | | | | |
| \$TESTN | 000404 | 1065 | 1113 | 1204* | 1503* | 1549 | 1573 | 1655 | 1699 | 1744 | 1832 | 1880 | 1968 | 2018 | | | | | | |
| | | 2110 | 2168 | 2176 | 2201 | 2236 | 2277 | 2378 | | | | | | | | | | | | |
| \$TN | 000014 | 963* | 973 | 1561 | 1574* | 1650 | 1656* | 1691 | 1700* | 1733 | 1745* | 1826 | 1833* | 1866 | | | | | | |
| | | 1881* | 1952 | 1969* | 2007 | 2019* | 2083 | 2111* | 2208 | 2237* | 2255 | 2278* | 2348 | 2379* | | | | | | |
| \$TPB | 177566 | 1150* | 1950* | | | | | | | | | | | | | | | | | |
| \$TPCHR | 006546 | 2937 | 2946* | 3037 | | | | | | | | | | | | | | | | |
| \$TPDEC | 006640 | 2517 | 2664 | 2699 | 2977* | | | | | | | | | | | | | | | |
| \$TPNUM | 007012 | 3027 | 3032* | | | | | | | | | | | | | | | | | |
| \$TPS | 177564 | 1157* | 2948 | | | | | | | | | | | | | | | | | |
| \$TPSTK | 005344 | 2572 | 2638* | | | | | | | | | | | | | | | | | |
| \$TSM | 000302 | 1107* | | | | | | | | | | | | | | | | | | |
| \$TYPE | 006504 | 1388 | 2932* | 2953 | 2959 | 2990 | 3020 | | | | | | | | | | | | | |
| \$UNIT | 000412 | 1059 | 1207* | | | | | | | | | | | | | | | | | |
| \$UNITM | 000306 | 1109* | | | | | | | | | | | | | | | | | | |
| \$USWR | 000424 | 1214* | | | | | | | | | | | | | | | | | | |
| \$Z | 000362 | 1193* | | | | | | | | | | | | | | | | | | |
| \$ZZ | 007744 | 2473* | | | | | | | | | | | | | | | | | | |
| \$SM | 000200 | 980* | 985* | 992 | 993* | 995* | 997* | 999* | 1005* | 1011* | 1050* | 1094 | 1095* | 1097* | | | | | | |
| | 007746 | 1099* | 1117* | 1121* | 1125* | 1129* | 1133* | 1135* | 1137* | 1142* | 1144* | 1147* | 1193 | 1196* | | | | | | |
| | | 1247* | 1256 | 1511 | 1574 | 1629 | 1658 | 1701 | 1745 | 1833 | 1882 | 1971 | 2020 | 2112 | | | | | | |
| | | 2238 | 2278 | 2379 | 2500 | 2757 | 2955* | 3245 | 3263* | 3305* | 3308* | | | | | | | | | |

.\$X = 000276 1094* 1099
 .ABS. 007746 000

ERRORS DETECTED: 0

DSKW: CZKMAP, CZKMAP/SOL/CRF/NL; TOC=CZKMAP_P11
 RUN-TIME: 10 10 .6 SECONDS
 RUN-TIME RATIO: 111/21=5.1
 CORE USED: 11K (21 PAGES)