

MJ11

11/70 MEM TST
CEMJADO

AH 7991D-MC

COPYRIGHT 73 79
FICHE 1 OF 1

MAY 1979

digital

MADE IN USA

.REM @

IDENTIFICATION

PRODUCT CODE: AC-7990D-MC
PRODUCT NAME: CEMJADO 11/70 MEM TST
PRODUCT DATE: 12-FEB-1979
MAINTAINER: DIAGNOSTIC ENGINEERING

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

NO RESPONSIBILITY IS ASSUMED FOR THE USE OR RELIABILITY OF SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL OR ITS AFFILIATED COMPANIES.

COPYRIGHT (C) 1973, 1979 BY DIGITAL EQUIPMENT CORPORATION

THE FOLLOWING ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION:

DIGITAL	PDP	UNIBUS	MASSBUS
DEC	DECUS	DECTAPE	

TABLE OF CONTENTS

1.0	ABSTRACT
2.0	REQUIREMENTS
2.1	EQUIPMENT
2.2	STORAGE
2.3	PRELIMINARY PROGRAMS
3.0	LOADING & STARTING PROCEDURE
3.1	ACT11 OPERATION
4.0	SWITCH SETTINGS
5.0	SUBROUTINE ABSTRACTS
5.1	SCOPE
6.0	ERRORS
6.1	PARITY ERROR
7.0	RESTRICTIONS
7.1	STARTING RESTRICTION
7.2	OPERATION RESTRICTION
8.0	MISCELLANEOUS
8.1	STACK POINTER
8.2	PASS COUNT
8.3	ERROR COUNT
8.4	DISPLAY REGISTER
8.5	POWER FAIL
8.6	EXECUTION TIME
9.0	PROGRAM DESCRIPTION

50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101

1.0 ABSTRACT

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

2.0 REQUIREMENTS

2.1 EQUIPMENT

PDP-11/70 FAMILY PROCESSOR WITH 32K MEMORY

2.2 STORAGE

PROGRAM STORAGE - THE PROGRAM USES MEMORY 0-17777

2.3 PRELIMINARY PROGRAMS

DEKBA THROUGH DEKBF

3.0 LOADING AND STARTING PROCEDURE

LOAD PROGRAM INTO MEMORY USING ABS LOADER

LOAD ADDRESS 200

SET SW12 IN DESIRED POSITION (SEE SEC 4.0)

PRESS START.

ASTERISK "*" WILL BE PRINTED AFTER EACH PASS.

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

PASS COUNT MAY BE MONITORED IN THE DISPLAY REGISTER.

NOTE: THIS PROGRAM SAVES THE LOADERS (BOOT AND ABS), TO RESTORE THE LOADERS, RESTART AT 162.

3.1 ACT11 OPERATION

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

4.0 SWITCH SETTINGS

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

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

SW14 = 1 OR UP.... LOOP SUBTEST

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

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

NOTE: INHIBITING THE USE OF MEMORY MANAGEMENT CAN

158
159

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

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

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

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

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

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

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

SW6 = 1 OR UP..... USE 18 BIT UNIBUS MAPPING ONLY
(TEST ONLY FIRST 128K OF MEMORY)

5.0 SUBROUTINE ABSTRACTS

5.1 SCOPE

THE PROGRAM STORES IN R1 THE PC OF THE LAST TEST SUCCESSFULLY
EXECUTED AND MAY BE USED AS AN AID IN DEBUGGING IF THE PROGRAM
'BOMBS' BECAUSE OF A HARDWARE FAILURE.

6.0 ERRORS

THESE TESTS PRINT OUT THE PC WHERE THE ERROR WAS DETECTED, THE
FAILING ADDRESS, THE GOOD DATA, AND THE BAD DATA I.E.

PC=XXXXXX ADDRESS AAAAAA GOOD DATA GGGGGG BAD DATA BBBBBB

THE ADDRESS OF THE FAILING LOCATION IS THE TRUE 22 BIT
PHYSICAL ADDRESS.

NOTE: WHEN TESTING MEMORY LOCATIONS 0-77776 THE PC TYPED WILL
BE A MULTIPLE OF 10000 GREATER THAN REFLECTED IN THE PROGRAM
LISTING

THE ADDRESS OF THE BAD DATA IS IN (R2) -2

THE GOOD DATA IN R0

THE BAD DATA IN R3

THE ADDRESS OF GOOD DATA IS IN R4 (RANDOM DATA TEST ONLY)

WHEN AN ERROR IS DETECTED WHEN EXERCISING THE MEMORY USING THE
WORST CASE NOISE PATTERNS, THE USER SHOULD RESTART THE PROGRAM
SELECTING PROGRAM #2(SEE SEC 9.1 FOR DETAILS) SELECTING THE
APPROPRIATE PARAMETERS. THE USER CAN USE THE PC AND ADDRESS
OF THE FAILURE TO SELECT THE PROPER CORE BANK(S) AFFECTED AND
ALSO THE SPECIFIC PATTERN. THIS ALLOWS MAXIMUM SCOPE
CAPABILITIES.

6.1 PARITY ERROR

CEMJADO 11/70 MEM TST MACY11 30A(1052) 09-FEB-79 12:56 PAGE 7
CEMJAD.P11 09-FEB-79 11:18

SEQ 0006

216
217
218
219

IF A PARITY ERROR IS DETECTED THE PROGRAM WILL TYPE:
PARITY ERROR
PC=PPPPPP MEMORY ADDRESS IS AAAAAAAAAA
PARITY ERROR REG=EEEEEE ?????????? MARGIN

WHERE PPPPPP IS THE CONTENTS OF THE PC WHEN THE PARITY ERROR OCCURRED, AAAAAAAA IS THE ADDRESS OF THE WORD, EEEEEEE IS THE CONTENTS OF THE MEMORY ERROR REGISTER, AND ?????????? IS THE MARGIN SETTING AT THE TIME OF THE PARITY ERROR.

AFTER REPORTING THE PARITY ERROR THE PROGRAM WILL START OVER.

7.0 RESTRICTIONS

7.1 STARTING RESTRICTION
PROGRAM MUST NOT BE RELOCATED WHEN RESTARTING

7.2 OPERATIONAL RESTRICTION
PROGRAM CHECKS CONTIGUOUS MEMORY IF A PARITY ERROR TRAP OCCURS WHEN THE PROGRAM IS RELOCATED PROGRAM ACTION IS UNDEFINED. IF PARITY MEMORY IS AVAILABLE OR SELECTED THE 3XOR9 TEST PATTERN IS FOR PARITY MEMORY ONLY. DO NOT POWER FAIL THE PROGRAM WHEN THE PROGRAM IS RUNNING RELOCATED.

8.0 MISCELLANEOUS

IF THE PROGRAM HALTS IN THE TRAP/INTERRUPT VECTOR AREA (0-1000), EXAMINE REGISTER 6 (THE STACK PTR). R6 CONTAINS THE ADDRESS WHERE THE PC OF THE INSTRUCTION THAT CAUSED THE TRAP ABORT IS STORED. SEE ALSO R1 (R1 SPECIFIES THE LAST TEST COMPLETED).

NOTE: THE PDP-11/70 WILL DISPLAY THE TRAP VECTOR ADDRESS+4 IN THE ADDRESS LIGHTS. THUS A TRAP TO 4 (BUS ERROR) WILL DISPLAY 10 IN THE ADDRESS LIGHTS.

8.1 STACK POINTER
THE STACK POINTER IS INITIALLY SET TO 520
AND IS RESET TO THIS VALUE AT THE START OF EACH SUBTEST.

8.2 PASS COUNT
SIX PASSES ARE REQUIRED FOR COMPLETION OF THIS PROGRAM; AT WHICH TIME AN "*" WILL BE PRINTED. THE PASS COUNT MAY BE OBSERVED BY TURNING THE SWITCH TO THE DISPLAY POSITION. (THE PASS COUNT IS ALSO STORED IN LOCATION 1000.) THE PASS COUNT SHOULD BE MONITORED IN THE EVENT THAT THE PROGRAM ENTERS AN UNDEFINED LOOP..BLANK 1

8.3 ERROR COUNT
EACH TIME AN ERROR OCCURS, THE ERROR COUNT IS INCREMENTED. THE ERROR COUNT CAN BE OBSERVED BY TURNING THE SWITCH TO THE DISPLAY POSITION AND SETTING SWITCH 9. (THE ERROR COUNT IS ALSO STORED IN LOCATION 1002.) THE PROGRAM WILL COUNT 17777(8) ERRORS; THE ERROR COUNT IS NOT INCREMENTED PAST THIS VALUE..BLANK 1

8.4 DISPLAY REGISTER
EITHER THE PASS COUNT OR THE ERROR COUNT IS DISPLAYED IN THE DISPLAY REGISTER. THE COUNT TO BE DISPLAYED IS CONTROLLED BY THE SETTING OF SWITCH 9..BLANK 1

276
277

8.5 POWER FAIL

278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333

THE PROGRAM MAY BE POWER FAILED WHEN RUNNING. WHEN THE POWER RETURNS THE PROGRAM WILL CONTINUE IN SEQUENCE.**CAUTION** DO NOT TURN POWER OFF/ON UNTIL THE MESSAGE 'POWER FAILED' HAS BEEN TYPED. THIS IS BECAUSE THE STACK MAY OVERFLOW.

8.6 EXECUTION TIME
EXECUTION TIME IS DEPENDENT ON THE AMOUNT OF MEMORY.

9.0 PROGRAM DESCRIPTION
THE PROGRAM VERIFIES EACH ADDRESS BY WRITING THE VALUE OF EACH ADDRESS INTO ITSELF STARTING AT LOCATION 20000 AND ENDING AT THE LAST LOCATION IN MEMORY. THE VALUE OF THE LAST LOCATION +2 IS TYPED ON THE TTY. NEXT THE VALUES WRITTEN ARE VERIFIED. TO COMPLETE THE ADDRESS TEST THE COMPLEMENT VALUE OF EACH MEMORY ADDRESS IS WRITTEN STARTING AT THE LAST MEMORY ADDRESS AND ENDING AT ADDRESS 20000. THE WRITTEN COMPLEMENT VALUES ARE THEN VERIFIED. THE NEXT PHASE OF TESTING INCLUDES READING, WRITING AND CHECKING MEMORY USING WORST CASE NOISE TEST PATTERN. A SUBTEST IS DEDICATED TO CHECKING THE PATTERN. THE TEST PROCEEDS BY EXERCISING EACH BANK OF MEMORY USING THE WORST CASE PATTERN. THE PROGRAM THEN CHECKS MEMORY USING RANDOM DATA (RANTST). THIS ROUTINE MOVES THE PROGRAM CODE THROUGHOUT MEMORY STARTING AT LOCATION 20000, AND RELOCATES THE DATA BY A 32(10) WORD OFFSET ON EACH SUBSEQUENT RELOCATION. I.E., FIRST RELOCATION IS TO 20000, NEXT IS TO 20100, THEN 20200, ETC. AFTER RELOCATION THE CODE MOVED IS CHECKED AGAINST THE ORIGINAL CODE (0-17776). WHEN THE RANDOM DATA TEST IS COMPLETE THE PROGRAM THEN SUCCESSIVELY ROTATES A 0 BIT (ROTO) AND A '1' BIT (ROT1) THROUGH ALL OF MEMORY. WHEN ALL TESTING IS COMPLETE THE PROGRAM INCREMENTS THE PASS COUNT (LOCATION 1000) AND RESTARTS BEGINNING WITH THE WORST CASE NOISE TESTS. AN ASTERISK (*) WILL BE TYPED ON COMPLETION OF EACH PASS, AND WHEN 6 PASSES HAVE BEEN COMPLETED THE PROGRAM WILL TYPE 'CEMJA DONE' AND RESTART THE PROGRAM BEGINNING WITH THE MEMORY ADDRESS TESTS.

@
.NLIST MD,MC
.LIST ME
.ABS
.TITLE CEMJADO 11/70 MEM TST
.SBTTL STARTING INST & DEFINITIONS
;COPYRIGHT 1973,1979 DIGITAL EQUIPMENT CORP., MAYNARD,MASS.
;THIS TEST CHECKS THAT ALL MEMORY ADDRESSES ARE UNIQUE USING ADDRESS TESTS
;AND CHECKS DATA RELIABILITY OF MEMORY USING WORST CASE NOISE TEST PATTERNS
;A RANDOM # PATTERN (PROGRAM CODE RELOCATED), A ROTATING 0 AND ROTATING
;1 PATTERN.
;LOADING AND STARING INSTRUCTIONS
;LOAD ADDRESS 200 AND START
;THIS PROGRAM ALSO RELOCATES THE ABS AND BOOT LOADERS TO ALLOW TESTING


```

334      ;OF MEMORY, TO RESTORE THE LOADERS RESTART AT 162.
335      ;
336      ;   STACK POINTER IS SET AT 500
337      ;   AN ASTERISK '*' WILL BE PRINTED ON COMPLETION OF EACH PASS, AND
338      ;   THE PROGRAM NAME WILL BE PRINTED WHEN TEST IS COMPLETE.
339      ;GENERAL REGISTER ASSIGNMENTS
340      0000C0      R0=%0
341      000001      R1=%1
342      000002      R2=%2
343      000003      R3=%3
344      000004      R4=%4
345      000005      R5=%5
346      000006      SP=%6
347      000007      PC=%7
348      000000      R10=%0
349      000001      R11=%1
350      000002      R12=%2
351      000003      R13=%3
352      000004      R14=%4
353      000005      R15=%5
354
355      ;STATUS REGISTER (PSW) BIT ASSIGNMENTS
356      000001      C=1      ;C BIT
357      000002      V=2      ;V BIT
358      000004      Z=4      ;Z BIT
359      000010      N=10     ;N BIT
360      000020      T=20     ;'T' BIT
361      000340      PRTY7=340 ;PRIORITY LEVEL 7
362      000200      PRTY4=200 ;PRIORITY LEVEL 4
363      000000      KM=000000 ;KERNEL MODE
364      040000      SM=040000 ;SUPERVISORY MODE
365      140000      UM=140000 ;USER MODE
366      000000      PKM=000000 ;PREVIOUS KERNEL MODE
367      010000      PSM=010000 ;PREVIOUS SUPERVISORY MODE
368      030000      PUM=030000 ;PREVIOUS USER MODE
369      004000      REG=004000 ;SELECT R10-R15
370
371      ;VECTOR ADDRESSES
372      000004      ERRVEC=4   ;ADDRESS OF ERROR VECTOR
373      000010      RESVEC=10  ;ADDRESS OF RESERVED INST. TRAP VECTOR
374      000014      TBITVEC=14 ;ADDRESS OF 'T' BIT TRAP VECTOR
375      000014      TRTVEC=14  ;ADDRESS OF 'TRACE' TRAP VECTOR
376      000014      BPTVEC=14  ;ADDRESS OF 'BREAKPOINT' TRAP VECTOR
377      000020      IOTVEC=20   ;ADDRESS OF IOT TRAP VECTOR
378      000024      PFVEC=24    ;ADDRESS OF POWER FAIL TRAP VECTOR
379      000030      EMTVEC=30   ;ADDRESS OF EMT VECTOR
380      000034      TRAPVEC=34  ;ADDRESS OF TRAP VECTOR
381      000060      TKVEC=60    ;ADDRESS OF TTY KEYBOARD INTERRUPT VECTOR
382      000064      TPVEC=64    ;ADDRESS OF TTY PRINTER INTERRUPT VECTOR
383      000240      PIRVEC=240   ;ADDRESS OF PIRQ VECTOR
384      000244      FPEVEC=244  ;ADDRESS OF FLOATING POINT INT. VECTOR
385      000250      MMVEC=250   ;ADDRESS OF MEM MGMT ERROR TRAP VECTOR
386
387      ;REGISTER ADDRESSES
388      177776      PSW=177776  ;ADDRESS OF STATUS REGISTER
389      177774      SLR=177774  ;ADDRESS OF STACK LIMIT REGISTER
    
```

```

390      177772      PIRQ=177772      ;ADDRESS OF PROGRAM INTERRUPT REQUEST
391      177770      UBREAK=177770    ;ADDRESS OF MICRO BREAK REGISTER
392      177746      CNTRL=177746     ;ADDRESS OF 11/70 MEMORY CONTROL REGISTER
393      177560      TKS=177560       ;ADDRESS OF KEYBOARD CSR
394      177562      TKB=177562       ;ADDRESS OF KEYBOARD BUFFER
395      177564      TPS=177564       ;ADDRESS OF TELEPRINTER CSR
396      177566      TPB=177566       ;ADDRESS OF TELEPRINTER BUFFER
397      177570      SWR=177570       ;ADDRESS OF CONSOL SWITCH REGISTER
398      177570      DISPLAY=177570    ;ADDRESS OF CONSOL DISPLAY REGISTER
399
400      ;INITIAL STACK POINTER SETTING
401      000500      STKPTR=500
402
403      ;MISCELLANEOUS BIT ASSIGNMENTS
404      000100      BIT15= 100
405      040000      BIT14= 040000
406      020000      BIT13= 020000
407      010000      BIT12= 010000
408      001000      BIT9= 001000
409      000400      BIT8= 000400
410      000100      BIT6= 000100
411
412      ;MEMORY MANAGEMENT REGISTER ADDRESS ASSIGNMENTS
413      177572      SRO=177572      ;ADDRESS OF MEM MGMT REGISTER SRO
414      177574      SR1=177574      ;" " " " " " SR1
415      177576      SR2=177576      ;" " " " " " SR2
416      172516      SR3=172516      ;ADDRESS OF MEM MGMT REGISTER SR3
417
418      172300      KIPDR0=172300    ;ADDRESS OF KERNEL 'I' PAGE
419      172302      KIPDR1=172302    ;DESCRIPTOR REGISTERS
420      172304      KIPDR2=172304
421      172306      KIPDR3=172306
422      172310      KIPDR4=172310
423      172312      KIPDR5=172312
424      172314      KIPDR6=172314
425      172316      KIPDR7=172316
426
427      172340      KIPAR0=172340     ;ADDRESSES OD KERNEL 'I' SPACE
428      172342      KIPAR1=172342     ;PAGE ADRESS REGISTERS
429      172344      KIPAR2=172344
430      172346      KIPAR3=172346
431      172350      KIPAR4=172350
432      172352      KIPAR5=172352
433      172354      KIPAR6=172354
434      172356      KIPAR7=172356
435
436
437      ;INSTRUCTION EQUATES
438      104400      HLT=TRAP
439      104000      SCOPE=EMT      ;SCOPE IS AN EMT TRAP
440
441      ;MISC. EQUATES
442      000006      RW=6      ;R/W BIT IN PDR REGISTERS
443      000000      UP=0      ;UP BIT IN PDR REGISTERS
444
445

```



```

446
447
448
449
450
451
452
453
454
455
456
457
458
459
460      000000      000000      . = 0
461      000000      000000      .WORD 0
462      000002      000000      .WORD 0
463
464      000004      001126      .WORD ERRTRP
465      000006      000002      .WORD RTI
466      000034      001204      . = TRAPVEC
467      000036      000340      .WORD ERROR
468      000046      000046      .WORD PRTY7
469      000046      004304      . = 46
470      000046      004304      $ENDAD
471
472      000052      040000      . = 52
473      000052      000100      40000
474      000100      000746      . = 100
475      000104      000207      CRLF: JSR R5,$SPRINT
476      000106      000000      $CRLF
477      000110      000000      RTS PC
478      000112      000000      RELFL: .WORD 0
479      000162      012706      SAVPC2: .WORD 0
480      000166      004767      . = 162
481      000172      000401      PONE: MOV #500,SP ; STARTING ADDRESS TO RELOCATE LOADERS.
482      000174      000200      JSR PC,$RLDR
483      000200      012706      HALT
484      000204      000137      BR PTWO
485      000250      000000      . = 200
486      000252      000000      PTWO: MOV #500,SP ; STARTING ADDRESS OF MEMORY TEST.
487      000252      000000      JMP @#START ; GO TO START OF TEST
488      000252      000000      . = 250
489      000252      000000      .WORD 0 ; MEMORY MANAGEMENT TRAP VECTOR.
490      000252      000000      .WORD 0
491
492
493      ; ROUTINE TO SAVE REGISTERS ON THE STACK
494      ; CALLED BY SAVE MACRO OR JSR PC,$SAVR
495      000254      012667      $SAVR: MOV (SP)+,1$ ; SAVE RETURN PC
496      000260      010546      MOV R5,-(SP)
497      000262      010446      MOV R4,-(SP)
498      000264      010346      MOV R3,-(SP)
499      000266      010246      MOV R2,-(SP)
500      000270      010146      MOV R1,-(SP)
501      000272      010046      MOV R0,-(SP)

```

```

502 000274 012707      MOV      (PC)+,PC      ;RETURN
503 000276 000000      1$:      0              ;CONTAINS RETURN ADDRESS
504
505 ;ROUTINE TO RESTORE REGISTERS SAVED ON THE STACK
506 ;CALLED BY RESTORE MACRO OR JSR PC,$RESTR
507 000300 012667 000016 $RESTR: MOV      (SP)+,1$      ;SAVE RETURN PC
508 000304 012600      MOV      (SP)+,R0
509 000306 012601      MOV      (SP)+,R1
510 000310 012602      MOV      (SP)+,R2
511 000312 012603      MOV      (SP)+,R3
512 000314 012604      MOV      (SP)+,R4
513 000316 012605      MOV      (SP)+,R5
514 000320 012707      MOV      (PC)+,PC      ;RETURN
515 000322 000000      1$:      0              ;CONTAINS RETURN ADDRESS
516
517 ;SBTTL POWER FAIL ROUTINE
518 ;=502
519 000502
520 ;POWER FAIL ROUTINE
521 ;THE POWER DOWN ROUTINE SAVES THE KEYBOARD STATUS,THE GENERAL REGISTERS
522 ; (R0-R5),AND MEM MGMT REGISTERS (KIPDR0-KIPDR7,KIPAR0-KIPAR7,SR3,SR2,SR0)
523 ;ON THE STACK AND SAVES THE STACK POINTER IN PFSTK BELOW.
524 PDWN: MOV      @#TKS,-(SP)      ;SAVE KEYBOARD STATUS
525 JSR      PC,$SAVR      ;GO SAVE REGISTERS ON THE STACK
526 TST      @#MMAVA      ;CHECK IF MEM MGMT IS AVAILABLE
527 BEQ      3$            ;BRANCH IF NOT AVAILABLE
528 MOV      @#SR0,-(SP)      ;SAVE SR0
529 MOV      @#SR2,-(SP)      ;SAVE SR2
530 MOV      @#SR3,-(SP)      ;SAVE SR3
531 MOV      #KIPDR0,R0      ;GET ADDRESS OF KIPDR0
532 MOV      #8.,R2
533 MOV      R2,R3
534 1$: MOV      (R0)+,-(SP)      ;SAVE KIPDR0-KIPDR7
535 SOB      R2,1$
536 MOV      #KIPAR0,R0      ;GET ADDRESS OF KIPAR0
537 2$: MOV      (R0)+,-(SP)      ;SAVE KIPAR0-KIPAR7
538 SOB      R3,2$
539 3$: MOV      SP,(PC)+      ;SAVE STACK PTR IN FOLLOWING LOCATION
540 PFSTK: .WORD      0          ;CONTAINS STACK PTR AFTER POWER FAIL
541 MOV      #PUP,@#PFVEC      ;SET POWER FAIL VECTOR TO PUP ROUTINE
542 HALT
543
544 ;POWER UP ROUTINE.
545 PUP: NOP
546 MOV      @#PFSTK,SP      ;SET STACK PTR
547 TST      MMAVA          ;CHECK IF MEM MGMT IS AVAILABLE
548 BEQ      4$
549 MOV      #KIPAR7+2,R0      ;GET ADDRESS OF KIPAR7+2
550 MOV      #8.,R2
551 MOV      R2,R3
552 1$: MOV      (SP)+,-(R0)      ;RESTORE KIPAR7-KIPAR0
553 SOB      R3,1$
554 MOV      #KIPDR7+2,R0      ;GET ADDRESS OF KIPDR7+2
555 2$: MOV      (SP)+,-(R0)      ;RESTORE KIPDR7-KIPDR0
556 SOB      R2,2$
557 MOV      (SP)+,@#SR3      ;RESTORE SR3
558 MOV      (SP)+,@#SR2      ;RESTORE SR2

```



```

558 000650 012637 177572      MOV      (SP)+, @#SRO      ;RESTORE SRO
559 000654 005767 004630      4$:      TST      PARAVA      ;CHECK IF PARITY REGISTERS ARE ENABLED
560 000660 001402              BEQ      5$      ;BRANCH IF NOT
561 000662 004767 004522      JSR      PC, .MAMF      ;GO ENABLE PARITY REGISTERS
562 000666              5$:              ;
563 000666 004767 177406      JSR      PC, $RESTR      ;RESTORE REGISTERS FROM STACK
564 000672 012637 177560      MOV      (SP)+, @#TKS
565 000676 012737 000502 000024  MOV      #PDWN, @#PFVEC      ;SET POWER FAIL TRAP TO PDWN ROUTINE
566 000704 005027              CLR      (PC)+
567 000706 000000      10$:      .WORD      0
568 000710 005267 177772      11$:      INC      10$      ;DELAY WAITING FOR TTY MOTOR
569 000714 100375              BPL      11$
570 000716 004567 000046      JSR      R5, $PRINT      ;GO TO PRINT ROUTINE
571 000722 000730      PWRFAIL
572 000724 000240      6$:      NOP
573 000726 000002      RTI      ;RETURN
574
575 000730 005015 047520 042527 PWRFAIL: .ASCII <15><12>'POWER FAILED'
576 000736 020122 040506 046111
577 000744 042105
578 000746 005015 000      $CRLF: .ASCIIZ <15><12>
579
580
581              .SBTTL      TAGS & PRINT ROUTINE
582              .EVEN
583 000752 000000      ICNT:      .WORD      0      ;CONTAINS PASS COUNT
584 000754 000000      ICOUNT: .WORD      0      ;CONTAINS ITERATION PATTERN
585 000756 000000      ERCNT: 0      ;CONTAINS ERROR COUNT
586 000760 000000      LDDISP: 0      ;CONTAINS DISPLAY REGISTER IMAGE
587 000762 000000      MMAVA: 0      ;MEM MGMT AVAILABLE INDICATOR
588              ;0=NOT AVAIL, -1=AVAIL(18 BIT MODE)
589              ;-2=AVAIL(22 BIT MODE)
590
591 000764 000000      RELOC: .WORD      0      ;CONTAINS RELOCATION FACTOR
592 000766 000000      COUNT: .WORD      0      ;TEMPORARY WORKING LOCATION
593
594      ;ROUTINE TO PASS MESSAGE ADDRESS TO TYPE ROUTINE BELOW
595      ;CALL: JSR      R5, $PRINT
596      ;      MESSAGE ADDRESS
597 000770 000240      $PRINT: NOP
598 000772 012567 000016      MOV      (R5)+, 1$      ;GET MESSAGE ADDRESS
599 000776 066767 177762 000010      ADD      RELOC, 1$      ;ADD RELOCATION FACTOR
600 001004 013746 177776      MOV      @#PSW, -(SP)      ;PUSH PSW ON THE STACK
601 001010 004767 000014      JSR      PC, .TYPE      ;CALL TYPE ROUTINE
602 001014 000000      1$:      .WORD      0      ;CONTAINS MESSAGE ADDRESS
603 001016 000205      RTS      R5      ;RETURN
604
605      ;ROUTINE TO TYPE ASCII MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
606      ;THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
607      ;CALL: TYPE
608      ;      MESADR      ;MESADR IS FIRST ADDRESS OF ASCIIZ STRING
609
610      ;TAGS USED BY THE TYPE ROUTINE BELOW
611 001020 000      $NULL: .BYTE      0      ;CONTAINS NULL CHARACTER
612 001021 002      $FILL: .BYTE      2      ;CONTAINS # OF FILLER CHARACTERS
613 001022 000      $TPFLG: .BYTE      0      ;CONTAINS TELEPRINTER AVAILABLE FLAG

```

Address	Op Code	Operand 1	Operand 2	Operand 3	Instruction	Comments
614	001023	000			\$TKFLG: .BYTE 0	:0/377 = AVAIL/NOT AVAIL
615	001024	177564			\$TPS: .WORD 177564	:CONTAINS KEYBOARD AVAILABLE FLAG
616	001026	177566			\$TPB: .WORD 177566	:ADDRESS OF TELEPRINTER STATUS REGISTER
617	001030	010046			.TYPE: MOV R0,-(SP)	:ADDRESS OF TELEPRINTER DATA BUFFER
618	001032	017600	000002		MOV @2(SP),R0	:SAVE R0
619	001036	062766	000002	000002	ADD #2,2(SP)	:GET MESSAGE ADDRESS
620						:ADJUST RETURN PC
621						
622	001044	112046			1\$: MOVB (R0)+,-(SP)	:PUSH CHARACTER TO BE TYPED ONTO STACK
623	001046	001003			BNE 2\$	:BRANCH IF NOT THE TERMINATOR
624	001050	005726			TST (SP)+	:POP TERMINATOR CHAR OFF THE STACK
625	001052	012600			MOV (SP)+,R0	:RESTORE R0
626	001054	000002			RTI	:RETURN TO CALLER
627						
628	001056	004767	000026		2\$: JSR PC,TYPIT	:TYPE CHARACTER
629	001062	122726	000012		3\$: CMPB #12,(SP)+	:CHECK IF CHARACTER WAS A LINE FEED
630	001066	001366			BNE 1\$	:BRANCH IF NOT LINE FEED
631	001070	016746	177724		MOV \$NULL,-(SP)	:GET # OF FILLERS REQUIRED AND FILLER CHARACTER.
632						
633						
634	001074	105366	000001		4\$: DECB 1(SP)	:DECREMENT FILLERS REQ. COUNT
635	001100	002770			BLT 3\$	:BRANCH IF NO MORE FILLERS ARE REQUIRED
636	001102	004767	000002		JSR PC,TYPIT	:TYPE FILLER CHARACTER
637	001106	000772			BR 4\$	
638						
639	001110	105777	177710		TYPIT: TSTB @ \$TPS	:WAIT FOR OUTPUT DEVICE
640	001114	100375			BPL -4	
641	001116	116677	000002	177702	MOVB 2(SP),@ \$TPB	:OUTPUT CHARACTER
642	001124	000207			RTS PC	
643						
644					:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	
645					:ERROR TRAP SERVICE ROUTINE	
646	001126	005737	177570		ERRTRP: TST @ \$SWR	:CHECK IF HALT ON ERROR
647	001132	100001			BPL +4	:BRANCH IF NO HALT ON ERROR
648	001134	000000			HALT	:HALT
649	001136	005727			TST (PC)+	:CHECK IF PREV TRAP TO 4 REPORTED
650	001140	000000			1\$: .WORD 0	:CONTAINS ERROR REPORTED FLAG
651	001142	001013			BNE 2\$	:BRANCH IF NOT REPORTED
652	001144	010667	177770		MOV SP,1\$	:SET 'NOT REPORTED'
653	001150	011602			MOV (SP),R2	:GET PC OFF STACK
654	001152	004767	000376		JSR PC,\$FORMO	:GO TO FORMAT ROUTINE
655	001156	004567	177606		JSR R5,\$PRINT	:GO TO PRINT ROUTINE
656	001162	001460			TRAP4	
657	001164	004567	177600		JSR R5,\$PRINT	:GO TO PRINT ROUTINE
658	001170	002351			DIGITS	
659	001172	000000			2\$: HALT	:ERROR! SECOND TRAP TO 4 OCCURRED BEFORE FIRST WAS PRINTED
660						
661	001174	005067	177740		CLR 1\$	
662	001200	000137	000200		JMP @ #200	:RESTART AT 200
663						
664					.SBTTL ERROR SERVICE ROUTINE	
665					:ERROR SERVICE CALLED BY JSR PC.ERROR INSTRUCTION	
666					:OR HLT (A TRAP INST)	
667	001204	000240			ERROR: NOP	
668	001206	022767	017777	177542	CMP #17777,ERCNT	:CHECK FOR MAX ERROR CNT
669	001214	001403			BEQ 4\$	

670	001216	062767	000001	177532		ADD	#1,ERCNT	;INCREMENT ERROR COUNT
671	001224	032737	001000	177570	4\$:	BIT	#BIT9,@#SWR	;SWITCH 9 UP?
672	001232	001411				BEQ	5\$	
673	001234	042767	017777	177516		BIC	#17777,LDDISP	;SAVE RELOCATION BITS
674	001242	056767	177510	177510		BIS	ERCNT,LDDISP	;LOAD ERROR COUNT
675	001250	016737	177504	177570		MOV	LDDISP,@#DISPLAY	;LOAD DISPLAY REGISTER
676	001256	005737	177570		5\$:	TST	@#SWR	;HALT ON ERROR
677	001262	100002				BPL	.+6	
678	001264	000000				HALT		
679	001266	000470				BR	3\$	
680	001270	032737	020000	177570		BIT	#20000,@#SWR	;PRINT OUT DESIRED?
681	001276	001051				BNE	1\$	;BRANCH IF NO PRINTOUT
682	001300	004767	176750			JSR	PC,\$SAVR	;GO SAVE REGISTERS ON THE STACK
683	001304	016602	000014			MOV	14(SP),R2	;GET PC OF ERROR CALL
684	001310	004767	000240			JSR	PC,\$FORMO	;GO TO FORMAT ROUTINE
685	001314	004567	177450			JSR	R5,\$PRINT	;GO TO PRINT ROUTINE
686	001320	001475				ERRPC		
687	001322	004567	177442			JSR	R5,\$PRINT	;GO TO PRINT ROUTINE
688	001326	002351				DIGITS		
689	001330	016602	000004			MOV	4(SP),R2	;GET FAILING ADDRESS (IN R2)
690	001334	004767	000214			JSR	PC,\$FORMO	;GO TO FORMAT ROUTINE
691	001340	004567	177424			JSR	R5,\$PRINT	;GO TO PRINT ROUTINE
692	001344	002327				ADRESS		
693	001346	105767	003221			TSTB	PENFLG	;BRANCH IF PARITY ERROR DETECTED
694	001352	001017				BNE	11\$	;BUT NOT FOUND
695	001354	105767	003212			TSTB	PEFLG	;BRANCH IF PARITY ERROR DETECTED
696	001360	001006				BNE	10\$	;BUT FOUND
697	001362	004567	177402			JSR	R5,\$PRINT	;GO TO PRINT ROUTINE
698	001366	001501				XMTDAT		
699	001370	010046				MOV	R0,-(SP)	;PUSH VALUE TO TYPED ONTO STACK
700	001372	004767	000416			JSR	PC,02A	;GO PRINT VALUE
701	001376				10\$:			
702	001376	004567	177366			JSR	R5,\$PRINT	;GO TO PRINT ROUTINE
703	001402	001514				RECDAT		
704	001404	010346				MOV	R3,-(SP)	;PUSH VALUE TO BE TYPED ONTO STACK
705	001406	004767	000402			JSR	PC,02A	
706	001412	004767	176462		11\$:	JSR	PC,CRLF	
707	001416	004767	176656			JSR	PC,\$RESTR	;RESTORE REGISTERS FROM STACK
708	001422	032737	002000	177570	1\$:	BIT	#2000,@#SWR	;RING BELL ON ERROR
709	001430	001403				BEQ	2\$	
710	001432	004567	177332			JSR	R5,\$PRINT	;GO TO PRINT ROUTINE
711	001436	001527				BELL		
712	001440	005737	177570		2\$:	TST	@#SWR	;HALT AFTER PRINT OUT
713	001444	100001				BPL	.+4	
714	001446	000000				HALT		
715	001450	010042			3\$:	MOV	R0,-(R2)	;RESTORE CORRECT DATA TO ADDRESS
716	001452	062702	000002			ADD	#2,R2	
717	001456	000002				RTI		
718								
719	001460	051124	050101	042520	TRAP4:	.ASCII	'TRAPPED TO 4 '	
720	001466	020104	047524	032040				
721	001474	040						
722	001475	120	036503	000	ERRPC:	.ASCII	'PC='	
723	001501	107	047517	020104	XMTDAT:	.ASCII	'GOOD DATA='	
724	001506	040504	040524	000075				
725	001514	041040	042101	042040	RECDAT:	.ASCII	'BAD DATA='	

726	001522	052101	036501	000		
727	001527	007	000			
728	001531	120	051101	052111	BELL: .ASCIZ <7>	
729	001536	020131	051105	047522	PARREG: .ASCIZ /PARITY ERROR REG=	
730	001544	020122	042522	036507		
731	001552	000				
732		001554				
733					.EVEN	
734	001554	066767	177204	000014	:ROUTINE TO PLACE ASCII VALUE OF AN ADDRESS IN TO ADDRESS MESSAGE	
735	001562	066767	177176	000152	\$FORM0: ADD RELOC,11\$+2	
736	001570	004767	176460		ADD RELOC,41\$+2	
737	001574	012704	002351		JSR PC,\$SAVR	;GO SAVE REGISTERS ON THE STACK
738	001600	005003			11\$: MOV #DIGITS,R4	;ADDRESS WHERE ASCII VALUES ARE STORED
739	001602	162702	000002		CLR R3	;WORKING & INDEX REGISTER
740	001606	010205			SUB #2,R2	;ADJUST ADDRESS
741	001610	010501			MOV R2,R5	;SAVE
742	001612	005767	177144		MOV R5,R1	
743	001616	001426			TST MMAVA	;CHECK IF MEM MGMT IS AVAILABLE
744	001620	032737	000001	177572	BEQ 1\$	;BRANCH IF NOT AVAILABLE
745	001626	001422			BIT #1,@#SRO	;IS MEM MGMT ENABLED
746	001630	042701	017777		BEQ 1\$	;BR IF NOT = ZERO
747	001634	000301			BIC #17777,R1	;SAVE PAR SELECTOR BITS
748	001636	006001			SWAB R1	;SWAP BYTES
749	001640	006001			ROR R1	
750	001642	006001			ROR R1	;FORM INDEX VALUE
751	001644	006001			ROR R1	
752	001646	017102	001774		MOV @PARTAB(1),R2	;GET CONTENTS OF PAR
753	001652	012700	000006		MOV #6,R0	;SHIFT COUNT
754	001656	006302			ASL R2	;SHIFT KIPAR1 6 PLACES LEFT
755	001660	006103			ROL R3	;MSB'S GO INTO R3
756	001662	077003			SOB R0,-4	;BR IF NOT = ZERO
757	001664	042705	160000		BIC #160000,R5	;CLEAR PAR SELECTOR BITS
758	001670	060502			ADD R5,R2	;FORM 22 BIT ADDRESS
759	001672	005503			ADC R3	;IN R2 & R3
760	001674	005001			1\$: CLR R1	
761	001676	012700	000005		MOV #5,R0	
762	001702	006003			12\$: ROR R3	
763	001704	006002			ROR R2	
764	001706	006001			ROR R1	
765	001710	005300			DEC R0	
766	001712	001373			BNE 12\$	
767	001714	012700	000010		MOV #8.,R0	;DIGIT COUNT
768	001720	000405			BR 3\$	;PRINT FIRST DIGIT
769	001722	006301			2\$: ASL R1	
770	001724	006102			ROL R2	
771	001726	006103			ROL R3	
772	001730	005305			DEC R5	
773	001732	001373			BNE 2\$	
774	001734	012705	000003		3\$: MOV #3,R5	;DIGIT SHIFT COUNT
775	001740	116324	002312		41\$: MOVB DIGTAB(3),(4)+	;LOAD DIGIT INTO MESSAGE
776	001744	005003			CLR R3	;CLEAR INDEX
777	001746	005300			DEC R0	;DEC DIGIT COUNT
778	001750	001364			BNE 2\$	
779	001752	004767	176322		JSR PC,\$RESTR	;RESTORE REGISTERS FROM STACK
780	001756	046767	177002	177612	BIC RELOC,11\$+2	
781	001764	046767	176774	177750	BIC RELOC,41\$+2	


```

782 001772 000207          RTS      PC          ;RETURN
783
784 001774 172340          PARTAB: KIPAR0
785 001776 172342          KIPAR1
786 002000 172344          KIPAR2
787 002002 172346          KIPAR3
788 002004 172350          KIPAR4
789 002006 172352          KIPAR5
790 002010 172354          KIPAR6
791 002012 172356          KIPAR7
792
793          ;ROUTINE TO TYPE OCTAL VALUE PUSHED ONTO STACK
794          ;CALL: MOV      VALUE,-(SP)      ;PUSH VALUE ONTO STACK
795          ;      JSR      PC,02A          ;CALL ROUTINE
796 002014
797 002014 004767 176234    02A:      JSR      PC,$SAVR      ;GO SAVE REGISTERS ON THE STACK
798 002020 016600 000016    MOV      16(SP),R0      ;GET VALUE
799 002024 012703 000006    MOV      #6,R3          ;COUNTER
800 002030 005002          CLR      R2          ;WORKING REGISTER
801 002032 006100          ROL      R0
802 002034 006102          ROL      R2
803 002036 062702 000260    1$:      ADD      #260,R2      ;FORM ASCII VALUE
804 002042 010267 000040    MOV      R2,2$      ;MOVE CHAR TO TYPE LOCATION
805 002046 004567 176716    JSR      R5,$PRINT      ;GO TO PRINT ROUTINE
806 002052 002106          2$
807 002054 005002          CLR      R2
808 002056 006100          ROL      R0
809 002060 006102          ROL      R2
810 002062 006100          ROL      R0
811 002064 006102          ROL      R2
812 002066 006100          ROL      R0
813 002070 006102          ROL      R2
814 002072 005303          DEC      R3
815 002074 001360          BNE      1$
816 002076 004767 176176    JSR      PC,$RESTR      ;RESTORE REGISTERS FROM STACK
817 002102 012616          MOV      (SP)+,(SP)
818 002104 000207          RTS      PC
819 002106 000000          2$:      .WORD    0          ;CONTAINS CHARACTER TO BE TYPED
820
821 002110 000000          LODFLO: .WORD    0
822          ;ROUTINE TO SAVE ABS LOADER
823 002112 005767 177772    $LDR:      TST      LODFLO
824 002116 001401          BEQ      3$
825 002120 000207          RTS      PC
826 002122 012700 017776    3$:      MOV      #17776,R0
827 002126 012737 002140 000004    MOV      #2$,@#ERRVEC      ;SET TIME OUT TRAP VECTOR
828 002134 005720          TST      (R0)+
829 002136 000776          BR      -2
830 002140 022626          2$:      CMP      (SP)+,(SP)+
831 002142 022700 020000    CMP      #20000,R0      ;4K MACHINE?
832 002146 001417          BEQ      -4$      ;YES--GET OUT
833 002150 162700 005672    SUB      #1500,+1*2,R0      ;POINT R0 BACK TO LOADER
834 002154 010067 000102    MOV      R0,$LDR1      ;SAVE FOR RESTORE ROUTINE
835 002160 012702 002734    MOV      #1500.,R2      ;WORD COUNT
836 002164 012703 010256    MOV      #LODAR,R3      ;WHERE LOADER IS TO BE STORED
837 002170 012023          1$:      MOV      (R0)+,(R3)+      ;STORE LOADER

```

```

838 002172 005302          DEC      R2
839 002174 001375          BNE      1$
840 002176 014367 000042    MOV      -(R3),LSTLOC ;SAVE LAST WORD OF LOADERS
841 002202 005367 177702    DEC      LODFLO
842 002206 000207          RTS      PC ;RETURN
843
844          ;ROUTINE TO RESTORE LOADER
845 002210 005767 177674    $RLDR:  TST      LODFLO
846 002214 001001          BNE      2$
847 002216 000207          RTS      PC
848 002220 016705 000036    2$:      MOV      $LDR1,R5 ;GET FIRST ADDRESS OF WHERE LOADER IS
849                                     ;TO BE RESTORED
850 002224 012704 010256    MOV      #LODAR,R4 ;ADDRESS WHERE LOADER IS STORED
851 002230 012702 002734    MOV      #1500.,R2 ;WORD COUNT
852 002234 012425          1$:      MOV      (R4)+,(R5)+ ;RESTORE
853 002236 005302          DEC      R2
854 002240 001375          BNE      1$
855 002242 012745          MOV      (PC)+,-(R5) ;RESTORE LAST LOCATION (SAVED BY SAVE
856 002244 000000          LSTLOC: .WORD 0 ;LOADERS ROUTINE ABOVE)
857 002246 004567 176516    JSR      R5,$PRINT ;GO TO PRINT ROUTINE
858 002252 002264          $LDRM
859 002254 005067 177630    CLR      LODFLO
860 002260 000207          RTS      PC ;RETURN TO CALLER
861
862 002262 000000          $LDR1: .WORD 0 ;FIRST ADDRESS WHERE LOADERS ARE TO BE
863                                     ;RESTORED TO
864 002264 047514 042101 051105 $LDRM: .ASCIZ 'LOADER IS RESTORED'<15><12>
865 002272 044440 020123 042522
866 002300 052123 051117 042105
867 002306 005015 000
868 002312          .EVEN
869          ;DIGIT TABLE
870 002312 030460          DIGTAB: '01
871 002314 031462          '23
872 002316 032464          '45
873 002320 033466          '67
874
875          ;MESSAGES
876 002322 040514 052123 040    LST: .ASCII 'LAST '
877 002327 115 046505 051117    ADRESS: .ASCII 'MEMORY ADDRESS IS '
878 002334 020131 042101 051104
879 002342 051505 020123 051511
880 002350 040
881 002351 060 030060 030060    DIGITS: .ASCII '00000000'
882 002356 030060 060
883 002361 040 000    SPACE1: .ASCIZ ' '
884 002363 120 051501 036523    PASSMG: .ASCII 'PASS='
885 002370 020040 000    PASSNM: .ASCIZ ' '
886 002374          .EVEN
887 002374 000000          PLACE: .WORD 0
888                                     .SBTTL MEMORY ADDRESS TESTS
889
890          ;THIS TEST ADDRESS MEMORY UP TO 128K AND PROVES 'UNIQUENESS' OF ALL
891          ;MEMORY ADDRESS IN A 32K SEGMENT. THE TEST WRITES INTO EACH MEMORY
892          ;ADDRESS THE VALUE OF THAT ADDRESS AND THEN CHECKS FOR THE CORRECT
893          ;DATA IN EACH ADDRESS.

```



```

894      ;THE TWELVE MOST SIGNIFICANT BITS OF THE LAST AVAILABLE MEMORY ADDRESS
895      ;IS STORED IN R5.
896      ;STARTING INSTRUCTIONS
897      :      LOAD ADDRESS=200
898      :      PRESS START
899      :      STACK POINTER IS AT 500
900      :*****RESTART AT 162 TO RESTORE LOADER*****
901      ;MEMORY ADDRESS TEST
902      002376 012737 002440 000212 START: MOV #START1,@#212 ;CHANGE START ADDRESS
903      002404 012706 000500      MOV #STKPTR,SP ;SET UP STACK PTR
904      002410 004767 177476      JSR PC,$LDR ;GO SAVE MONITOR & LOADERS
905      002414 004567 176350      JSR R5,$PRINT ;GO TO PRINT ROUTINE
906      002420 007564      RESLDR
907      002422 005037 000756      CLR @#ERCNT ;CLEAR ERROR COUNT
908      002426 005037 000760      CLR @#LDDISP ;CLEAR DISPLAY REGISTER STORAGE LOCN
909      002432 013737 000760 177570      MOV @#LDDISP,@#DISPLAY ;CLEAR DISPLAY REGISTER
910      002440 012706 000500      START1: MOV #STKPTR,SP ;SET STACK PTR
911      002444 005037 004572      CLR @#PEFLG ;CLEAR PARITY ERROR INDICATORS
912      002450 052737 000014 177746      BIS #14,@#CNTRL ;DISABLE CACHE
913      002456 012727 002440      MOV #START1,(PC)+ ;LOAD PARITY ERROR RESTART ADDRESS
914      002462 000000      PERSTRT: .WORD 0 ;CONTAINS RESTART ADDRESS AFTER PAR ERR
915      002464 005037 000752      CLR @#ICNT ;CLEAR PASS COUNT
916      002470 005037 000764      CLR @#RELOCF ;CLEAR RELOCATION FACTOR
917      002474 012737 000502 000024      MOV #PDWN,@#PFVEC ;SET POWER FAIL TRAP VECTOR
918      002502 005037 000026      CLR @#PFVEC+2
919
920      ;CHECK IF MEMORY MANAGEMENT IS AVAILABLE
921      002506 005067 176250      CLR MMAVA ;CLEAR MEM MGMT AVAILABLE INDICATOR
922      002512 032737 010000 177570      BIT #BIT12,@#SWR ;CHECK IF TO RUN WITH MEM MGMT
923      002520 001034      BNE 1$ ;DO NOT USE MEM MGMT IF SW12 WAS SET
924      002522 012737 002612 000004      MOV #1$,@#ERRVEC ;SET TIME OUT TRAP
925      002530 005037 177572      CLR @#SRO ;REFERENCE MEM MGMT
926      002534 005167 176222      COM MMAVA ;SET INDICATOR TO -1 IF AVAILABLE
927      002540 032737 000100 177570      BIT #BIT6,@#SWR ;TEST MEM WITH 18 BIT MODE?
928      002546 001410      BEQ 2$ ;NO, USE 22 BIT MODE
929      002550 012737 000040 172516      MOV #40,@#SR3 ;ENABLE UNIBUS MAP
930      002556 022737 000040 172516      CMP #40,@#SR3 ;DID IT SET?
931      002564 001012      BNE 1$ ;NO, BRANCH
932      002566 000413      BR WRTUP ;NEXT TEST
933      002570 012737 000020 172516 2$: MOV #20,@#SR3 ;SET 22 BIT MODE
934      002576 022737 000020 172516      CMP #20,@#SR3 ;DID IT SET?
935      002604 001002      BNE 1$ ;NO, BRANCH
936      002606 006367 176150      ASL MMAVA ;YES--SET INDICATOR TO -2
937      002612 004767 002572 1$: JSR PC, .MAMF ;GO ENABLE PARITY ACTION
938
939
940      ;ROUTINE TO WRITE VALUE OF MEMORY ADDRESS INTO MEMORY ADDRESS
941      ;FOR EXAMPLE ROUTINE WRITES 20000 INTO LOCATION 20000
942      002616 012737 002656 000004 WRTUP: MOV #DONE0,@#ERRVEC ;SET TIME OUT TRAP VECTOR
943      002624 010701      MOV PC,R1 ;LOAD TRACE REGISTER
944      002626 004767 002706      JSR PC,LDMMO
945      002632 012737 005662 000250      MOV #MMABT0,@#MMVEC ;SET MEM MGMT ABORT VECTOR
946      002640 012702 020000      MOV #20000,R2 ;FIRST ADDRESS
947      002644 010203      MOV R2,R3 ;LOAD CONSTANT
948      002646 010322 1$: MOV R3,(R2)+ ;WRITE VALUE OF ADDRESS INTO ADDRESS
949      002650 062703 000002      ADD #2,R3 ;NEXT VALUE

```

```

950 002654 000774          BR      1$          ;WRITE UNTIL DONE
951
952 002656 012706 000500    DONE0: MOV    #STKPTR,SP      ;SET STACK PTR
953 002662 004767 176666    JSR     PC,$FORM0        ;GO TO FORMAT ROUTINE
954 002666 004567 176076    JSR     R5,$PRINT        ;GO TO PRINT ROUTINE
955 002672 002322          LST
956 002674 004767 175200    JSR     PC,CRLF
957
958          ;ROUTINE TO CHECK THAT VALUE OF MEMORY ADDRESS WAS WRITTEN CORRECTLY
959 002700 010701          MOV    PC,R1          ;LOAD TRACE REGISTER
960 002702 012702 020000    MOV    #20000,R2        ;SET R2
961 002706 012737 002762 000004    MOV    #DONE1,@#ERRVEC ;SET TIME OUT TRAP
962 002714 010200          MOV    R2,R0
963 002716 162700 000002    SUB     #2,R0          ;SUBTRACT 2
964 002722 004767 002612    JSR     PC,LDMMO
965 002726 062700 000002    1$:  ADD     #2,R0
966 002732 012203          MOV    (R2)+,R3        ;GET WRITTEN VALUE
967 002734 020003          CMP     R0,R3          ;CHECK
968 002736 001402          BEQ     2$
969 002740 104400          HLT
970 002742 000771          BR      1$
971 002744 005142    2$:  COM     -(R2)
972 002746 005112          COM     (R2)
973 002750 012203          MOV    (R2)+,R3
974 002752 020003          CMP     R0,R3
975 002754 001764          BEQ     1$
976 002756 104400          HLT
977          ;WRITTEN IMPROPERLY EXAMINE R2. NEXT EXAMINE MEM MGMT REGISTER KIPAR1
978          ; (IF MEM MGMT IS AVAILABLE). ADD R2 AND KIPAR1 TOGETHER AS SHOWN BELOW
979
980          :      R2-2          0 00x xxx xxx xxx xxx
981          :      KIPAR1(772342) Y YYY YYY YYY YYY YYY
982          :      ADDRESS      Z ZZZ ZZZ ZZZ ZZZ ZZZ ZZZ ZZZ
983
984 002760 000762          BR      1$
985 002762 012706 000500    DONE1: MOV    #STKPTR,SP      ;SET STACK PTR
986 002766 010701          MOV    PC,R1          ;LOAD TRACE REGISTER
987
988          ;ROUTINE TO WRITE 1'S COMPLEMENT VALUE OF ADDRESS INTO ADDRESS
989          ;FOR EXAMPLE ROUTINE WRITES 157777 INTO ADDRESS 20000
990
991 002770 005767 175766    TST     MAVA          ;MEMORY MAGNAGEMENT AVAILABLE?
992 002774 001420          BEQ     3$          ;BRANCH IF NOT USED
993 002776 013703 172342    MOV    @#KIPAR1,R3      ;FIND LAST ADDRESS IF MEM MANAGE USED
994 003002 006303          ASL     R3
995 003004 006303          ASL     R3
996 003006 006303          ASL     R3
997 003010 006303          ASL     R3
998 003012 006303          ASL     R3
999 003014 006303          ASL     R3
1000 003016 010246          MOV    R2,-(SP)        ;DEVELOP COMPLEMENT OF LAST ADDRESS
1001 003020 042716 020000    BIC     #20000,(SP)      ;SAVE BITS IF MEMORY IS NOT A MULTIPLE OF 4K
1002 003024 062603          ADD     (SP)+,R3
1003 003026 012737 005714 000250    MOV    #MMAB11,@#MMVEC ;SET ABORT VECTOR
1004 003034 000403          BR      2$
1005 003036 162702 000002    3$:  SUB     #2,R2          ;R2=LAST ADDRESS

```



```

1006 003042 010203
1007 003044 005103
1008 003046 062703 000002
1009 003052 010342
1010 003054 102403
1011 003056 020227 017776
1012 003062 001371
1013
1014 ;SET UP TO CHECK COMPLEMENT DATA WRITTEN DOWN
1015 DONE3: NOP
1016 003066 010701 MOV PC,R1 ;LOAD TRACE REGISTER
1017 003070 005767 175666 TST MMAVA ;CHECK IF MM IS AVAIL
1018 003074 001406 BEQ 1$
1019 003076 012737 000200 172342 MOV #200,@#KIPAR1 ;INIT KIPAR1
1020 003104 012737 005662 000250 MOV #MMABTO,@#MMVEC ;SET ABORT VECTOR
1021 003112 012737 003152 000004 1$: MOV #DONE4,@#ERRVEC
1022 003120 012702 020000 MOV #20000,R2 ;FIRST ADDRESS
1023 003124 010200 MOV R2,R0
1024 003126 005100 COM R0 ;FIRST DATA (COM OF ADDRESS)
1025 003130 062700 000002 ADD #2,R0
1026 003134 162700 000002 2$: SUB #2,R0
1027 003140 012203 MOV (R2)+,R3 ;GET VALUE
1028 003142 020003 CMP R0,R3 ;CHECK
1029 003144 001773 BEQ 2$
1030 003146 104400 HLT
1031 003150 000771 BR 2$
1032 003152 000240 DONE4: NOP
1033
1034 ;ROUTINE TO WRITE BANK # INTO ALL ADDRESSES IN A 4K BANK
1035 003154 012737 003222 000004 MOV #DONE4A,@#ERRVEC;SET TIME OUT TRAP VECTOR
1036 003162 010701 MOV PC,R1
1037 003164 004767 002350 JSR PC,LDMMO
1038 003170 012737 005662 000250 MOV #MMABTO,@#MMVEC
1039 003176 012702 020000 MOV #20000,R2
1040 003202 005000 CLR R0
1041 003204 005200 1$: INC R0 ;R0 WILL BE DATA WRITTEN
1042 003206 012704 010000 MOV #4096,R4 ;SET 4K COUNTER
1043 003212 010022 2$: MOV R0,(R2)+ ;WRITE BANK # INTO ALL ADDRESSES
1044 003214 005304 DEC R4
1045 003216 001375 BNE 2$
1046 003220 000771 BR 1$
1047
1048 003222 022626 DONE4A: CMP (SP)+,(SP)+ ;ADJUST STACK PTR
1049
1050 ;CHECK THAT DATA WRITTEN ABOVE CAN BE READ
1051 003224 012737 003272 000004 MOV #DONE4B,@#ERRVEC
1052 003232 010701 MOV PC,R1
1053 003234 004767 002300 JSR PC,LDMMO
1054 003240 012702 020000 MOV #20000,R2
1055 003244 005000 CLR R0
1056 003246 005200 1$: INC R0
1057 003250 012704 010000 MOV #4096,R4
1058 003254 012203 2$: MOV (R2)+,R3
1059 003256 020003 CMP R0,R3
1060 003260 001401 BEQ .+4
1061 003262 104400 HLT

```

1062	003264	005304			DEC	R4	
1063	003266	001372			BNE	2\$	
1064	003270	000766			BR	1\$	
1065	003272	022626			DONE4B: CMP	(SP)+, (SP)+	
1066							
1067					;ROUTINE TO WRITE CONSTANT DATA INTO 4K		
1068					;BANK STARTING WITH LAST MEMORY LOCATION		
1069	003274	010701			MOV	PC, R1	
1070	003276	012737	005714	000250	MOV	#MMABT1, @MMVEC	
1071	003304	162702	000002		SUB	#2, R2	
1072	003310	005000			CLR	R0	
1073	003312	005300			1\$: DEC	R0	
1074	003314	012704	010000		MOV	#4096., R4	
1075	003320	010042			2\$: MOV	R0, -(R2)	
1076	003322	102406			BVS	DONE4C	
1077	003324	020227	017776		CMP	R2, #17776	;CHECK IF DONE
1078	003330	001403			BEQ	DONE4C	
1079	003332	005304			DEC	R4	
1080	003334	001371			BNE	2\$	
1081	003336	000765			BR	1\$	
1082							
1083	003340	012737	003452	000004	DONE4C: MOV	#DONE4D, @ERRVEC	
1084	003346	010701			MOV	PC, R1	
1085	003350	004767	002164		JSR	PC, LDMMO	
1086	003354	012737	005662	000250	MOV	#MMABT0, @MMVEC	;SET ABORT VECTOR
1087	003362	012702	020000		MOV	#20000, R2	
1088	003366	022704	010000		1\$: CMP	#4096., R4	;CHECK IF WRITE ABOVE STARTED ON ;4K BOUNDARY
1089							
1090	003372	001415			BEQ	2\$	
1091	003374	012203			MOV	(R2)+, R3	
1092	003376	020003			CMP	R0, R3	
1093	003400	001402			BEQ	4\$	
1094	003402	104400			HLT		
1095	003404	000406			BR	5\$	
1096	003406	005142			4\$: COM	-(R2)	
1097	003410	005112			COM	(R2)	
1098	003412	012203			MOV	(R2)+, R3	
1099	003414	020003			CMP	R0, R3	
1100	003416	001401			BEQ	5\$	
1101	003420	104400			HLT		
1102	003422	005204			5\$: INC	R4	
1103	003424	001360			BNE	1\$	
1104	003426	005200			2\$: INC	R0	
1105	003430	012704	010000		MOV	#4096., R4	
1106	003434	012203			3\$: MOV	(R2)+, R3	
1107	003436	020003			CMP	R0, R3	
1108	003440	001401			BEQ	.+4	
1109	003442	104400			HLT		
1110	003444	005304			DEC	R4	
1111	003446	001372			BNE	3\$	
1112	003450	000766			BR	2\$	
1113							
1114	003452	022626			DONE4D: CMP	(SP)+, (SP)+	
1115	003454	005737	000042		TST	@#42	;BRANCH IF PROGRAM WAS NOT
1116	003460	001406			BEQ	BEGIN1	;LOADED VIA ACT11 IN QV OR AA MODES
1117	003462	005767	000620		TST	\$ENDAD+2	;BRANCH IF NOT IN QV MODE


```

1118 003466 100003      BPL      BEGIN1
1119 003470 012737 000001 004236      MOV      #1,@#ENDCT      ;SET ENDCT TO DO 1 PASS ONLY IN QV
1120      .SBTTL      WORST CASE NOISE TESTS
1121      ;THIS TEST WRITES MEMORY WORST CASE NOISE TEST PATTERNS THROUGHOUT
1122      ;MEMORY AND CHECKS THAT THEY CAN BE WRITTEN AND READ.
1123      ;SET UP TRAP VECTORS
1124 003476 012706 000500      BEGIN1: MOV      #STKPTR,SP      ;SET STACK PTR
1125 003502 052737 000014 177746      BIS      #14,@#CNTRL      ;DISABLE CACHE
1126 003510 004767 001674      JSR      PC,,MAMF      ;GO ENABLE PARITY ACTION
1127 003514 004767 003742      JSR      PC,CKSWR      ;GO CHECK SWITCHES
1128 003520 005027      CLR      (PC)+      ;SET INDICATOR TO WRITE NORMAL 3X9 PAT
1129 003522 000000      PARPAT: .WORD      0
1130 003524 005767 175232      TST      MMAVA      ;18 OR 22 BIT MODE?
1131 003530 001402      BEQ      DONE6      ;NO--BRANCH
1132 003532 004767 001754      JSR      PC,MARGIN      ;YES--GO SETUP MARGINS
1133
1134
1135      ;WRITE 3 XOR 9 TEST PATTERN STARTING AT ADDRESS 20000
1136      ;NOTE PATTERN IS NORMAL 3 XOR 9 IF NO PARITY MEMORY IS AVAILABLE,
1137      ;AND IS A MODIFIED PATTERN IF PARITY MEMORY IS AVAILABLE.
1138      ;THE CONTENTS OF PARPAT IF 0/NOT 0 INDICATE IF NORMAL/MODIFIED PATTERN
1139      ;IS BEING USED IN TESTS BELOW.
1140 003536 012706 000500      DONE6: MOV      #STKPTR,SP      ;SET STACK PTR
1141 003542 010701      MOV      PC,R1      ;UPDATE TRACE REGISTER
1142 003544 012737 003564 000004      MOV      #DONE7,@#ERRVEC      ;SET TIME OUT TRAP VECTOR
1143 003552 012746 000001      MOV      #1,-(SP)      ;PUSH STARTING BANK # ON STACK
1144 003556 005046      CLR      -(SP)      ;PUSH # OF 256. WORD BLOCKS TO WRITE
1145 003560 004767 002344      JSR      PC,.3X9      ;CALL ROUTINE TO WRITE 3XOR9 PATTERN
1146
1147      ;CHECK 3 XOR 9 TEST PATTERN WRITTEN ABOVE
1148 003564 012737 001126 000004      DONE7: MOV      #ERRTRP,@#ERRVEC
1149 003572 016600 000006      MOV      6(SP),R0      ;GET # OF 256. WORD BLOCKS WRITTEN
1150 003576 005400      NEG      R0      ;FORM TWO'S COMPLEMENT
1151 003600 010027      MOV      R0,(PC)+      ;SAVE # OF 256 WORD BLOCKS
1152 003602 000000      WDS.256: .WORD      0      ;CONTAINS # OF 256 WORD BLOCKS IN MEM.
1153 003604 012706 000500      MOV      #STKPTR,SP      ;SET STACK PTR
1154 003610 010701      MOV      PC,R1      ;SET SCOPE PTR
1155 003612 012746 000001      MOV      #1,-(SP)      ;PUSH BANK # ON THE STACK
1156 003616 010046      MOV      R0,-(SP)      ;PUSH # OF 256. WORD BLOCKS TO WRITE
1157 003620 004767 002524      JSR      PC,.3X9      ;GO CHECK DATA WRITTEN
1158
1159      ;SETUP TO RUN MODIFIED 3 XOR 9 PATTERN IF PARITY MEMORY IS AVAILABLE
1160 003624 005767 175132      TST      MMAVA      ;18 OR 22 BIT MODE?
1161 003630 001403      BEQ      1$      ;NO--BRANCH
1162 003632 005737 005510      TST      @#PARAVA      ;BRANCH IF PARITY MEMORY IS NOT AVAIL
1163 003636 001406      BEQ      DONE8
1164 003640 005737 003522      1$: TST      @#PARPAT      ;BRANCH IF PARITY PAT JUST WRITTEN
1165 003644 001003      BNE      DONE8
1166 003646 010637 003522      MOV      SP,@#PARPAT      ;SET INDICATOR TO WRITE 3X9 PAR PAT
1167 003652 000731      BR      DONE6      ;REPEAT TEST USING MODIFIED 3X9 PATTERN
1168
1169      ;WRITE 8 XOR 13 TEST PATTERN STARTING AT ADDRESS 40000
1170 003654 012706 000500      DONE8: MOV      #STKPTR,SP      ;SET STACK PTR
1171 003660 012737 003702 000004      MOV      #DONE9,@#ERRVEC      ;SET TIME OUT TRAP VECTOR
1172 003666 010701      MOV      PC,R1      ;UPDATE TRACE REGISTER
1173 003670 012746 000002      MOV      #2,-(SP)      ;PUSH STARTING BANK # ON THE STACK

```

```

1174 003674 005046          CLR      -(SP)          ;PUSH # OF BANKS TO WRITE ON THE STACK
1175 003676 004767 003242    JSR      PC,.8X13      ;GO TO ROUTINE TO WRITE DATA
1176
1177                          ;CHECK 8 XOR 13 TEST PATTERN WRITTEN ABOVE
1178 003702 012706 000500    DONE9:  MOV      #STKPTR,SP      ;SET STACK PTR
1179 003706 010701          MOV      PC,R1          ;UPDATE TRACE REGISTER
1180 003710 012737 001126 000004    MOV      #ERRTRP,@#ERRVEC
1181 003716 012746 000002    MOV      #2,-(SP)
1182 003722 005404          NEG      R4
1183 003724 042704 000001    BIC      #1,R4          ;SET 4K BANK COUNT TO 8K INCREMENT
1184 003730 001403          BEQ      DONE10         ;DO NOT CHECK IF ONLY 12K
1185 003732 010446          MOV      R4,-(SP)
1186 003734 004767 003212    JSR      PC,..8X13      ;GO CHECK 8 XOR 13 PATTERN WRITTEN ABOVE
1187
1188
1189 003740 000005          DONE10: RESET          ;DISABLE MEM MGMT AND PARITY ACTION
1190
1191
1192                          .SBTTL  RANDOM DATA,ROTATING I/O TESTS
1193                          ;RANDOM DATA TEST. THIS TEST MOVES THE PROGRAM CODE THROUGHOUT MEMORY
1194 003742 010701          RANTST:  MOV      PC,R1          ;SET TRACE POINTER
1195 003744 012737 004102 000004    MOV      #7,@#ERRVEC      ;SET TIME OUT TRAP
1196 003752 005767 175004          TST      MMAVA          ;CHECK IF MEM MGMT IS AVAILABLE
1197 003756 001412          BEQ      1$          ;BRANCH IF NOT AVAILABLE
1198 003760 004767 001554          JSR      PC,LDMMO          ;GO SET UP MEM MGMT
1199 003764 105237 172301          INCB     @#KIPDR0+1      ;ALLOW 4K ADDRESSING IN FIRST 4K
1200 003770 012737 077406 172304    MOV      #200*256.-400+UP+RW,@#KIPDR2 ;SET KIPDR2=RW UP 200 BLOCKS
1201 003776 012737 000400 172344    MOV      #400,@#KIPAR2
1202 004004 012702 020000    1$:  MOV      #20000,R2      ;SET 'TO' ADDRESS POINTER
1203 004010 005004          CLR      R4          ;SET 'FROM' ADDRESS POINTER
1204 004012 012705 004000    2$:  MOV      #2048.,R5      ;SET 4K WORD COUNT
1205 004016 012422    3$:  MOV      (R4)+,(R2)+      ;MOVE CODE
1206 004020 012422          MOV      (R4)+,(R2)+
1207 004022 005305          DEC      R5          ;DECREMENT 4K WORD COUNTER
1208 004024 001374          BNE      3$
1209
1210 004026 012705 005405          MOV      #4096.-PLACE+1,R5      ;SET 4K WORD COUNTER
1211 004032 014400    4$:  MOV      -(R4),R0      ;GET 'GOOD' DATA
1212 004034 014203          MOV      -(R2),R3      ;GET 'BAD' DATA
1213 004036 020003          CMP      R0,R3          ;COMPARE 'GOOD' & 'BAD' DATA
1214 004040 001403          BEQ      5$
1215 004042 005722          TST      (R2)+          ;STEP ADDRESS FOR ERROR ROUTINE
1216 004044 104400          HLT          ;REPORT ERROR
1217 004046 005742          TST      -(R2)          ;RESTORE ADDRESS POINTER
1218 004050 005305    5$:  DEC      R5          ;DECREMENT 4K WORD COUNTER
1219 004052 001367          BNE      4$          ;LOOP UNTIL 4K WORDS CHECKED
1220
1221 004054 005767 174702          TST      MMAVA          ;CHECK IF MEM MGMT IS AVAILABLE
1222 004060 001405          BEQ      6$          ;BRANCH IF NOT AVAILABLE
1223 004062 005237 172342          INC      @#KIPAR1
1224 004066 005237 172344          INC      @#KIPAR2
1225 004072 000744          BR      1$
1226 004074 062702 000100    6$:  ADD      #64.,R2          ;STEP ADDRESS
1227 004100 000744          BR      2$
1228 004102 012706 000500    7$:  MOV      #STKPTR,SP      ;RESET STACK PTR
1229 004106 012737 001126 000004    MOV      #ERRTRP,@#ERRVEC ;RESTORE ERROR TRAP VECTOR

```



```

1230
1231
1232 004114 012767 177777 003304 ;ROTATING 0 TEST. THIS TEST ROTATES A SINGLE '0' THROUGH MEMORY
1233 004122 012746 000001 ROTO: MOV #-1,.CONST ;SET CONSTANT =177777
1234 004126 016746 177450 MOV #1,-(SP) ;SET BANK #1
1235 004132 004767 003252 MOV WDS.256,-(SP) ;GET # OF 256. WORD BLOCKS IN MEMORY
1236 004136 0107C1 JSR PC,WRTPAT ;GO WRITE 1'S THROUGHOUT MEMORY
1237 004140 012746 000001 MOV PC,R1 ;SET SCOPE PTR
1238 004144 016746 177432 MOV #1,-(SP) ;SET STARTING BANK #
1239 004150 004767 003004 MOV WDS.256,-(SP) ;SET # OF 256. WORD BLOCKS TO CHECK
1240 JSR PC,.ROTO ;GO TO ROTATE 0 ROUTINE
1241
1242 ;ROTATING 1 TEST THIS TEST ROTATES A SINGLE '1' BIT THROUGH ALL OF
1243 004154 005067 003246 ;MEMORY
1244 004160 012746 000001 ROT1: CLR .CONST ;CLEAR CONSTANT
1245 004164 016746 177412 MOV #1,-(SP) ;PUSH STARTING BANK ONTO STACK
1246 004170 004767 003214 MOV WDS.256,-(SP) ;AND # OF 256. WORD BLOCKS IN MEMORY
1247 004174 010701 JSR PC,WRTPAT ;GO WRITE 0'S THROUGHOUT MEMORY
1248 004176 012746 000001 MOV PC,R1 ;SET SCOPE PTR
1249 004202 016746 177374 MOV #1,-(SP) ;SET STARTING BANK #
1250 004206 004767 003042 MOV WDS.256,-(SP) ;SET # OF 256. WORD BLOCKS TO CHECK
1251 JSR PC,.ROT1 ;GO ROTATE A '1' BIT THROUGHOUT MEMORY
1252
1253 004212 000005 ;END OF CYCLE
1254 004214 042737 000014 177746 END: RESET ;RESET MACHINE TO KEY-START STATE
1255 004222 010701 BIC #14,@#CNTRL ;UPDATE TRACE REGISTER
1256 004224 012706 000500 MOV PC,R1 ;SET STACK PTR
1257 004230 005237 000752 MOV #STKPTR,SP ;INCREMENT PASS COUNT
1258 004234 022737 INC @#ICNT ;CHECK FOR LAST PASS
1259 004236 000006 ENDCT: .WORD 6 ;MAKE 5 PASSES
1260 004240 000752 .WORD ICNT ;PASS COUNT ADDRESS
1261 004242 001405 BEQ DONE ;BRANCH IF LAST PASS COMPLETED
1262 004244 004567 174520 JSR R5,$PRINT ;GO TO PRINT ROUTINE
1263 004250 010152 ASTERISK
1264 004252 00013 003476 JMP @#BEGIN1
1265 004256 DONE: JSR R5,$PRINT ;GO TO PRINT ROUTINE
1266 004256 004567 174506 ENDMSG
1267 004262 010154 TSTB @#TPS ;WAIT FOR BELL TO RING
1268 004264 105737 177564 BPL -4
1269 004270 100375 MOV @#42,R0 ;GET DECTAPE MONITOR RETURN ADDRESS
1270 004272 013700 000042 BEQ FINISH
1271 004276 001400 JSR PC,$RLDR ;RESTORE MONITOR & LOADERS
1272 004300 004767 175704 $ENDAD: JSR PC,(R0) ;GO TO DECTAPE MONITOR
1273 004304 004710 NOP
1274 004306 000240 NOP
1275 004310 000240 NOP
1276 004312 000240 FINISH: JMP START1
1277 004314 000167 176120
1278
1279 .SBTTL PROGRAM SUBROUTINES
1280 .SBTTL RELOCATION ROUTINES
1281 ;ROUTINE TO RELOCATE PROGRAM CODE
1282 004320 012500 RELOC: MOV (R5)+,R0 ;GET FROM ADDRESS
1283 004322 011502 MOV (R5),R2 ;GET TO ADDRESS
1284 004324 010203 MOV R2,R3
1285 004326 062703 017776 ADD #17776,R3 ;MOVES 4K

```

```

1286 004332 012737 004402 000004      MOV    #4$,@#ERRVEC      ;SET TIME OUT TRAP
1287 004340 005004                      CLR    R4              ;CLEAR RELOCATION SUCCESSFUL INDICATOR
1288 004342 005723                      TST    (R3)+            ;CHECK IF MEMORY IS AVAILABLE
1289 004344 012022      1$:      MOV    (R0)+,(R2)+        ;RELOCATE
1290 004346 020203                      CMP    R2,R3          ;RELOCATION COMPLETE?
1291 004350 001375                      BNE    1$
1292 004352 011503                      MOV    (R5),R3
1293 004354 020203      2$:      CMP    R2,R3
1294 004356 001413                      BEQ    5$              ;BRANCH IF DONE
1295 004360 024042                      CMP    -(R0),-(R2)      ;CHECK THAT DATA WAS RELOCATED PROPERLY
1296 004362 001774                      BEQ    2$
1297 004364 005703                      TST    R3              ;CHECK IF RELOCATING BACK TO 000000
1298 004366 001403                      BEQ    3$
1299 004370 104400                      HLT
1300                                ;ERROR! CANNOT RELOCATE PROGRAM CODE
1301                                ;TO UPPER MEMORY BANK PROPERLY
1301 004372 000000                      HALT
1302 004374 000767                      BR     2$              ;CONTINUE RELOCATING AT YOUR PERIL
1303 004376 000000      3$:      HALT
1304                                ;ERROR! CANNOT RELOCATE CODE BACK TO
1305                                ;TO 000000 PROPERLY
1305 004400 000777                      BR
1306 004402 022626      4$:      CMP    (SP)+,(SP)+        ;RESTORE STACK PTR
1307 004404 005104                      COM    R4
1308 004406 000240      5$:      NOP
1309 004410 012702 000764      MOV    #RELOC,F,R2      ;GET ADDRESS OF RELOCATION FACTOR
1310 004414 061502                      ADD    (R5),R2        ;ADD FACTOR
1311 004416 012512                      MOV    (R5)+,(R2)      ;RELOCATED RELOCF NOW CONTAINS RELOCATION
1312                                ;FACTOR
1313 004420 000205                      RTS     5              ;RETURN, R4=-1 IF NO RELOCATION
1314
1315
1316                                ;ROUTINE TO RELOCATE PROGRAM CODE FROM ORIGINAL POSITION (0-4K) TO
1317                                ;TOP OF MEMORY.
1318 004422 012700 020000 000004      RELOCP: MOV    #20000,R0      ;SET UP TO SCAN FOR TOP OF MEMORY
1319 004426 012737 000006 000004      MOV    #ERRVEC+2,@#ERRVEC
1320 004434 062700 020000      1$:      ADD    #20000,R0      ;INCREMENT SCAN ADDRESS
1321 004440 000261                      SEC
1322 004442 005710                      TST    (R0)            ;SET TIME OUT INDICATOR
1323 004444 103373                      BCC    1$              ;CHECK FOR EXISTANT MEMORY
1324 004446 012737 001126 000004      MOV    #ERRTRP,@#ERRVEC  ;'C' WILL BE CLEAR IF MEMORY EXISTS
1325 004454 162700 020000                      SUB    #20000,R0      ;ADJUST TO LAST EXISTANT 4K

```


1326	004460	010067	000006	MOV	R0,28	;PASS RELOCATION ADDRESS TO RELOC ROUTINE
1327	004464	004567	177630	JSR	R5,RELOC	;RELOCATE PROGRAM

CEMJADO 11/70 MEM TST MACV11 30A(1052) 09-FEB-79 12:56 PAGE 30
CEMJAD.P11 09-FEB-79 11:18 RELOCATION ROUTINES

SEQ 0029

1328	004470	000000	
1329	004472	000000	
1330	004474	004567	174270
1331	004500	010113	

28:	000000	
	.WORD	0
	JSR	R5,\$PRINT
	RELOC	

:FROM ADDRESS 000000
:TO LAST 4K BANK
:GO TO PRINT ROUTINE


```

1332 004502 016746 177764      MOV      2$,-(SP)      ;PASS TO 02A ROUTINE
1333 004506 062716 010170      ADD      #REL24K,(SP)  ;SET UP RESTART ADDRESS
1334 004512 004767 175276      JSR      PC,02A        ;TYPE RESTART ADDRESS
1335 004516 011667 000006      MOV      (SP),3$      ;SAVE RETURN ADDRESS IN 3$ BELOW
1336 004522 066706 177744      ADD      2$,SP        ;RESET STACK PTR
1337 004526 012716             MOV      (PC)+,(SP)    ;GET RETURN ADDRESS
1338 004530 000000             .WORD      0              ;CONTAINS RETURN PC
1339 004532 066716 177734      ADD      2$,(SP)      ;ADJUST RETURN PC
1340 004536 000207             RTS      PC

1341
1342
1343
1344
1345
1346 004540 010067 000172      .SBTTL  MA/MF PARITY ERROR SERVICE ROUTINE
1347 004544 012700 004740      ;WHEN MA/MF A PARITY ERROR IS DETECTED THIS ROUTINE SCANS MEMORY FOR THE
1348 004550 010120             ;ADDRESS CAUSING THE PARITY ERROR. WHEN THE ADDRESS IS LOCATED THE ROUTINE
1349 004552 010220             ;HALTS WITH THE ADDRESS+2 IN R0. TO CONTINUE AFTER THE ERROR PRESS CONTINUE.
1350 004554 010320             .PARSRV:MOV      R0,SAVR0      ;SAVE R0 IN SAVR0
1351 004556 010420             MOV      #SAVR0+2,R0
1352 004560 010520             MOV      R1,(R0)+
1353 004562 004567 174202      MOV      R2,(R0)+
1354 004566 004752             MOV      R3,(R0)+
1355 004570 005027             MOV      R4,(R0)+
1356 004572 000             MOV      R5,(R0)+
1357 004573 000             JSR      R5,$PRINT      ;GO TO PRINT ROUTINE
1358 004574 012737 004642 000114  PARERR
1359 004602 012737 004702 000004  CLR      (PC)+      ;CLEAR PARITY ERROR INDICATORS
1360 004610 005002             PEFLG: .BYTE      0          ;NOT 0/0 =PAR ERR/NO PAR ERR
1361 004612 005767 174144      PENFLG: .BYTE      0          ;NOT 0/0=PAR ERR DETECTED/NOT DETECTED ON SCAN
1362 004616 001407             MOV      #2$,@#PARVEC      ;SET PARITY ERROR TRAP
1363 004620 004767 000714             MOV      #4$,@#ERRVEC      ;SET TIME OUT TRAP VECTOR
1364 004624 105237 172301             CLR      R2
1365 004630 012737 005662 000250      TST      MMAVA      ;CHECK IF MEM MGMT IS AVAILABLE
1366 004636 012200             BEQ      1$              ;BRANCH IF NOT AVAILABLE
1367 004640 000776             JSR      PC,LDMMO      ;SET UP MEM MGMT
1368 004642 110667 177724             INCB      @#KIPDR0+1      ;ALLOW FULL 4K PAGE ADDRESSING
1369 004646 010003             MOV      #MMABTO,@#MMVEC      ;SET MEM MGMT ABORT TRAP VECTOR
1370 004650 104400             1$: MOV      (R2)+,R0      ;SCAN ALL ADDRESSES
1371 004652 000177 003374             BR      1$
1372 004656 000240             2$: MOV      SP,PEFLG      ;SET PARITY ERROR FOUND INDICATOR
1373 004660 005067 177706             MOV      R0,R3
1374 004664 012706 000500             HLT
1375 004670 000005             JMP      @PARCLR      ;PARITY ERROR! ADDRESS+2 IS IN R2
1376 004672 004767 000512             3$: NOP
1377 004676 000177 003342             CLR      PEFLG      ;MUST CLEAR BAD PAR LOCATION
1378
1379
1380 004702 105767 177664             MOV      #STKPTR,SP      ;INSERT HALT INST TO EXAMINE PARITY REGS
1381 004706 001363             RESET      ;CLEAR PARITY ERROR INDICATORS
1382 004710 016602 000004             JSR      PC,.MAMF      ;RESET STACK PTR
1383 004714 162702 000002             JMP      @CLRPAR      ;GO ENABLE PARITY ERROR DETECTION
1384 004720 110667 177647             ;CLR PAR ERROR LOCATION
1385 004724 004567 174040      ;SERVICE ROUTINE IF PARITY ERROR NOT DETECTED ON SCAN
1386 004730 004773             4$: TSTB      PEFLG      ;BRANCH IF PARITY ERROR WAS
1387 004732 104400             BNE      3$              ;DETECTED ON SCAN
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500

```

```

1388 004734 000750          BR      3$
1389                      ;THE BELOW 6 WORDS CONTAINS THE SAVED CONTENTS OF R0-R5 WHEN THE
1390                      ;PARITY ERROR OCCURRED
1391 004736 000000          SAVR0: .WORD 0
1392 004740 000000          SAVR1: .WORD 0
1393 004742 000000          SAVR2: .WORD 0
1394 004744 000000          SAVR3: .WORD 0
1395 004746 000000          SAVR4: .WORD 0
1396 004750 000000          SAVR5: .WORD 0
1397
1398 004752 005015 040520 044522 PARERR: .ASCIZ <15><12>'PARITY ERROR'<15><12>
1399 004760 054524 042440 051122
1400 004766 051117 005015 000
1401 004773 116 052117 043040 NOFIND: .ASCIZ 'NOT FOUND ON SCAN'<15><12>
1402 005000 052517 042116 047440
1403 005006 020116 041523 047101
1404 005014 005015 000
1405 005020          .EVEN
1406
1407
1408          177740          MEMLO=177740
1409          177742          MEMHI=177742
1410          177744          MEMERR=177744
1411
1412
1413 005020 005767 177546          .22PAR: TST      PEFLG          ;BEEN HERE BEFORE
1414 005024 001403          BEQ      1$          ;BRANCH IF NO
1415 005026 000000          HALT          ;YES -- DOUBLE PARITY ERROR
1416 005030 000177 175426          JMP      @PERSTR
1417 005034 010667 177532          1$: MOV      SP,PEFLG          ;SET PARITY ERROR FLAG
1418 005040 005737 177570          TST      @#SWR          ;HALT ON ERROR?
1419 005044 100001          BPL      100$          ;BRANCH IF NO
1420 005046 000000          HALT          ;YES
1421 005050 013746 177744          100$: MOV      @#MEMERR,-(SP) ;SAVE MEMORY ERROR REG
1422 005054 013701 177740          MOV      @#MEMLO,R1 ;GET ADDRESS OF WHERE THE PARITY
1423 005060 013702 177742          MOV      @#MEMHI,R2 ;ERROR OCCURRED
1424 005064 011637 177744          MOV      (SP),@#MEMERR ;CLEAR THE ERROR REG
1425 005070 032737 020000 177570 BIT      #BIT13,@#SWR ;INHIBIT ERROR TIMEOUT
1426 005076 001071          BNE      101$          ;BRANCH IF YES
1427          ;PRINT 'PARITY ERROR'
1428 005100 004567 173664          JSR      R5,$PRINT
1429 005104 004752          PARERR
1430          ;PRINT 'PC=XXXXXX'
1431 005106 004567 173656          JSR      R5,$PRINT
1432 005112 001475          ERRPC
1433 005114 016646 000002          MOV      2(SP),-(SP) ;GET PC AT TIME OF PARITY ERROR
1434 005120 066716 173640          ADD      RELOC,(SP)
1435 005124 004767 174664          JSR      PC,02A
1436 005130 004567 173634          JSR      R5,$PRINT
1437 005134 002361          SPACE1
1438          ;CHANGE 22-BIT ADDRESS TO OCTAL-ASCII!
1439 005136 012700 002351          MOV      #DIGITS,R0
1440 005142 012704 000010          MOV      #8.,R4
1441 005146 012705 000003          2$: MOV      #3.,R5
1442 005152 005003          3$: CLR      R3
1443 005154 006301          4$: ASL      R1

```


1444	005156	106102			ROLB	R2	
1445	005160	006103			ROL	R3	
1446	005162	077504			SOB	R5,4\$	
1447	005164	116320	002312		MOVB	DIGTAB(R3),(R0)+	
1448	005170	077412			SOB	R4,2\$	
1449					:PRINT	"MEMORY ADDRESS IS AAAAAAAA"	
1450	005172	004567	173572		JSR	R5,\$PRINT	
1451	005176	002327			ADRESS		
1452	005200	004767	172674		JSR	PC,CRLF	
1453					:PRINT	"PARITY ERROR REG=XXXXXX"	
1454	005204	004567	173560		JSR	R5,\$PRINT	
1455	005210	001531			PARREG		
1456	005212	011605			MOV	(SP),R5	
1457	005214	004767	174574		JSR	PC,02A	
1458	005220	004767	173544		JSR	PC,\$PRINT	
1459	005224	002361			SPACE1		
1460					:PRINT	THE MARGIN SETTING	
1461	005226	016700	173520		MOV	ICNT,R0	
1462	005232	116000	005532		MOVB	MRGNTB(R0),R0	
1463	005236	062700	005266		ADD	#MARTBL,R0	
1464	005242	011067	000004		MOV	(R0),5\$	
1465	005246	004567	173516		JSR	R5,\$PRINT	
1466	005252	005266			5\$: MARTBL		
1467	005254	004567	173510		JSR	R5,\$PRINT	
1468	005260	005375			MARMSG		
1469	005262	000177	175174		101\$: JMP	@PERSTRT	
1470					:MARGIN MESSAGE	TABLE	
1471	005266	005304			MARTBL: NORMAL		
1472	005270	000000			0		
1473	005272	005313			ESTRB		
1474	005274	005330			LSTRB		
1475	005276	005344			LCRNT		
1476	005300	005360			HCRNT		
1477	005302	005304			NORMAL		
1478							
1479					:MARGIN MESSAGES		
1480	005304	047516	046522	046101	NORMAL: .ASCIZ	'NORMAL'	
1481	005312	000					
1482	005313	105	051101	054514	ESTRB: .ASCIZ	'EARLY STROBE'	
1483	005320	051440	051124	041117			
1484	005326	000105					
1485	005330	040514	042524	051440	LSTRB: .ASCIZ	'LATE STROBE'	
1486	005336	051124	041117	000105			
1487	005344	047514	020127	052503	LCRNT: .ASCIZ	'LOW CURRENT'	
1488	005352	051122	047105	000124			
1489	005360	044510	044107	041440	HCRNT: .ASCIZ	'HIGH CURRENT'	
1490	005366	051125	042522	052116			
1491	005374	000					
1492	005375	040	040515	043522	MARMSG: .ASCIZ	' MARGIN'<12><15>	
1493	005402	047111	006412	000			
1494		005410			.EVEN		
1495							
1496					:ROUTINE TO ENABLE PARITY ERROR	ACTION ON 11/70 PARITY MEMORIES	
1497		000114			PARVEC=114	:PARITY ERROR INTERRUPT VECTOR ADDRESS	
1498							
1499	005410	032737	000040	177570	.MAMF: BIT	#40,@#SWR	:CHECK IF PARITY ERROR DETECTION IS TO

```

1500 005416 001007      BNE 1$ ;BE ENABLED. BRANCH IF NOT TO BE ENABLED
1501 ;ENABLE PARITY ERROR DETECTION
1502 005420 042737 000002 177746 BIC #2,@#CNTRL ;OTHERWISE, INSURE THAT PARITY ERROR
1503 ;DETECTION IS ENABLED
1504 005426 012767 000001 000054 MOV #1,PARAVA ;SET PARITY ERROR DETECTION INDICATOR
1505 005434 000405 BR 2$
1506 ;DISABLE PARITY ERROR DETECTION
1507 005436 052737 000002 177746 1$: BIS #2,@#CNTRL ;DISABLE PARITY ERROR DETECTION
1508 005444 005067 000040 CLR PARAVA ;CLEAR PARITY ERROR DETECTION INDICATOR
1509 ;SET-UP PARITY ERROR SERVICE TRAP FOR 18-BIT OR 22-BIT
1510 ;ADDRESSING MODES
1511 005450 012737 004540 000114 2$: MOV #.PARSRV,@#PARVEC ;SET-UP 18-BIT ADDRESS PARITY
1512 ;ERROR TRAP VECTOR
1513 005456 012737 000340 000116 MOV #340,@#PARVEC+2 ;PRIORITY LEVEL 7 ON TRAP
1514 005464 005767 173272 TST MMAVA ;18 OR 22 BIT MODE?
1515 005470 001406 BEQ 3$ ;BRANCH IF NOT, OTHERWISE
1516 005472 012737 005020 000114 MOV #.22PAR,@#PARVEC ;SET-UP 22-BIT ADDRESS PARITY
1517 ;ERROR TRAP VECTOR
1518 005500 012737 000340 000116 MOV #340,@#PARVEC+2 ;PRIORITY LEVEL 7 ON TRAP
1519 005506 000207 3$: RTS PC ;RETURN
1520 005510 000000 PARAVA: .WORD 0 ;PARITY ERROR DETECTION INDICATOR
1521 ;0 - PARITY ERROR DETECTION IS DISABLED
1522 ;1 - PARITY ERROR DETECTION IS ENABLED
1523
1524
1525 ;.SBTTL MARGIN ROUTINE
1526 ;ROUTINE TO SET THE MARGINS
1527 177750 MAINTRG=177750
1528
1529 005512 016700 173234 MARGIN: MOV 1CNT,R0 ;PASS COUNT
1530 005516 005002 CLR R2 ;FAST COUNTER
1531 005520 116037 005532 177750 MOVB MRGNTB(R0),@#MAINTRG ;LOAD MAINTENANCE REG.
1532 005526 077201 1$: SOB R2,1$
1533 005530 000207 RTS PC
1534
1535 005532 000 MRGNTB: .BYTE 0 ;NORMAL
1536 005533 004 .BYTE 4 ;EARLY STROBE
1537 005534 006 .BYTE 6 ;LATE STROBE
1538 005535 010 .BYTE 10 ;LOW CURRENT
1539 005536 012 .BYTE 12 ;HIGH CURRENT
1540 005537 000 .BYTE 0 ;NORMAL
1541
1542
1543 ;.SBTTL MEM MGMT ROUTINES
1544 ;ROUTINE TO INITIALIZE MEMORY MANAGEMENT REGISTERS
1545 005540 000240 LDMMO: NOP
1546 005542 005767 173214 TST MMAVA
1547 005546 001444 BEQ 1$
1548 005550 032737 000100 177570 BIT #BIT6,@#SWR ;18 BIT UNIBUS MAPPING?
1549 005556 001004 BNE 2$ ;YES--BRANCH
1550 005560 012737 000020 172516 MOV #20,@#SR3 ;22 BIT MODE
1551 005566 000403 BR 3$
1552 005570 012737 000040 172516 2$: MOV #40,@#SR3 ;ENABLE UNIBUS MAP
1553 005576 012737 077006 172300 3$: MOV #177*256.-400+UP+RW,@#KIPDR0 ;SET KIPDR0=RW UP 177 BS
1554 005604 012737 077406 172302 MOV #200*256.-400+UP+RW,@#KIPDR1 ;SET KIPDR1=RW UP 200 BLOCKS
1555 005612 005037 172304 CLR @#KIPDR2
    
```



```

1556 005616 005037 172344 CLR @#KIPAR2
1557 005622 012737 077406 172316 MOV #200*256.-400+UP+RW,@#KIPDR7 ;SET KIPDR7=RW UP 200 BLOCKS
1558 005630 005037 172340 CLR @#KIPAR0
1559 005634 012737 000200 172342 MOV #200,@#KIPAR1
1560 005642 012737 177600 172356 MOV #177600,@#KIPAR7
1561 005650 012737 000001 177572 MOV #1,@#SR0 ;ENABLE MEM MGMT
1562 005656 000240 NOP
1563 005660 000207 1$: RTS PC
1564
1565 ;MEMORY MANAGEMENT ABORT ROUTINE FOR WRITE UP
MMABT0: MOV #20000,R2 ;RESET R2
1566 005662 012702 020000 ADD #200,@#KIPAR1 ;ADVANCE TO NEXT 4K
1567 005666 062737 000200 172342 MOV @#SR2,(SP) ;RETURN TO INSTRUCTION THAT
1568 005674 013716 177576 CLR @#SR0 ;DISABLE MEM MGMT
1569 005700 005037 177572 MOV #1,@#SR0 ;ENABLE MEM MGMT
1570 005704 012737 000001 177572 RTI ;CAUSED THE ABORT
1571 005712 000002
1572
1573 ;MEM MGMT ABORT SERVICE FOR WRITE DOWN
MMABT1: MOV #40000,R2 ;RESET R2
1574 005714 012702 040000 SUB #200,@#KIPAR1
1575 005720 162737 000200 172342 BEQ 2$
1576 005726 001406 MOV @#SR2,(SP)
1577 005730 013716 177576 MOV #1,@#SR0 ;ENABLE MEM MGMT
1578 005734 012737 000001 177572 RTI
1579 005742 000002
1580 005744 2$: CLR @#SR0 ;DISABLE MEM MGMT
1581 005744 005037 177572 BIS #V,2(SP)
1582 005750 052766 000002 000002 RTI
1583 005756 000002
1584
1585 ;ROUTINE TO SET UP MEMORY MANAGEMENT FOR PATTERN TESTS
STMM2: TST R2 ;CHECK IF TESTING BANK # 0
1586 005760 005702 BEQ 2$ ;EXIT IF BANK # 0
1587 005762 001442 TST MMAVA
1588 005764 005767 172772 BNE 1$ ;BRANCH IF MEM MGMT AVAILABLE
1589 005770 001005 ROR R2 ;ADJUST ADDRESS
1590 005772 006002 ROR R2
1591 005774 006002 ROR R2
1592 005776 006002 ROR R2
1593 006000 006002 ROR R2
1594 006002 000207 RTS PC ;RETURN
1595
1596 006004 004767 177530 1$: JSR PC,LDMM0 ;GO MAKE INITIAL SET UP
1597 006010 000302 SWAB R2
1598 006012 006002 ROR R2
1599 006014 010237 172344 MOV R2,@#KIPAR2
1600 006020 062702 000200 ADD #200,R2
1601 006024 010237 172346 MOV R2,@#KIPAR3
1602 006030 012737 077406 172304 MOV #200*256.-400+UP+RW,@#KIPDR2 ;SET KIPDR2=RW UP 200 BLOCKS
1603 006036 012737 077406 172306 MOV #200*256.-400+UP+RW,@#KIPDR3 ;SET KIPDR3=RW UP 200 BLOCKS
1604 006044 005037 172310 CLR @#KIPDR4
1605 006050 012702 040000 MOV #40000,R2
1606 006054 012737 006072 000250 MOV #MMABT2,@#MMVEC
1607 006062 012737 000001 177572 MOV #1,@#SR0 ;ENABLE MEM MGMT
1608 006070 000207 2$: RTS PC
1609
1610 ;ROUTINE TO SERVICE 8 XOR 13 ABORTS
1611 006072 000240 MMABT2: NOP

```

1612	006074	012702	040000		MOV	#40000,R2	
1613	006100	062737	000400	172344	ADD	#400,@#K1PAR2	
1614	006106	062737	000400	172346	ADD	#400,@#K1PAR3	
1615	006114	013716	177576		MOV	@#SR2,(SP)	;SET RETURN TO INSTRUCTION THAT ABORTED
1616	006120	012737	000001	177572	MOV	#1,@#SR0	;ENABLE MEM MGMT
1617	006126	000002			RTI		
1618							
1619					.SBTTL	3 XOR 9 ROUTINES	
1620					;ROUTINE TO WRITE 3XOR9 WORST CASE NOISE TEST PATTERN		
1621					:CALL:	MOV BANK #,-(SP)	;PUSH STARTING BANK # ON STACK
1622					:	MOV BLKCNT,-(SP)	;PUSH 256. WORD BLOCK COUNT ON STACK
1623					:	JSR PC,,3X9	;CALL ROUTINE
1624							
1625	006130	016602	000004		.3X9:	MOV 4(SP),R2	;GET STARTING BANK #
1626	006134	004767	177620			JSR PC,STMM2	
1627	006140	005000				CLR R0	
1628	006142	010003				MOV R0,R3	
1629	006144	005103				COM R3	;R0 (0) AND R3 (-1) IS THE DATA WRITTEN
1630	006146	005767	175350			TST PARPAT	;BRANCH IF PARITY MEMORY PATTERN IS
1631	006152	001402				BEQ 1\$	;NOT TO BE WRITTEN
1632							
1633	006154	012700	000401			MOV #401,R0	;WRITE PARITY 3X9 PATTERN
1634	006160	012704	000020		1\$:	MOV #16.,R4	;EACH LOOP WRITES 256. WORDS
1635							
1636	006164	010022			2\$:	MOV R0,(R2)+	
1637	006166	010022				MOV R0,(R2)+	
1638	006170	010022				MOV R0,(R2)+	
1639	006172	010022				MOV R0,(R2)+	
1640							
1641	006174	010022				MOV R0,(R2)+	
1642	006176	010022				MOV R0,(R2)+	
1643	006200	010022				MOV R0,(R2)+	
1644	006202	010022				MOV R0,(R2)+	
1645							
1646	006204	010322				MOV R3,(R2)+	
1647	006206	010322				MOV R3,(R2)+	
1648	006210	010322				MOV R3,(R2)+	
1649	006212	010322				MOV R3,(R2)+	
1650							
1651	006214	010322				MOV R3,(R2)+	
1652	006216	010322				MOV R3,(R2)+	
1653	006220	010322				MOV R3,(R2)+	
1654	006222	010322				MOV R3,(R2)+	
1655							
1656	006224	005304				DEC R4	
1657	006226	001356				BNE 2\$	
1658	006230	005100				COM R0	
1659	006232	005103				COM R3	
1660	006234	005767	175262			TST PARPAT	;BRANCH IF PARITY MEMORY PATTERN IS
1661	006240	001402				BEQ 3\$	;NOT TO BE WRITTEN
1662							
1663	006242	004767	000014			JSR PC,,XOR39	;GO GET CONSTANTS
1664	006246	005366	000002		3\$:	DEC 2(SP)	;DECREMENT 256. WORD BLOCK COUNT
1665	006252	001342				BNE 1\$	
1666	006254	012616				MOV (SP)+,(SP)	;ADJUST STACK
1667	006256	012616				MOV (SP)+,(SP)	

	RTS	PC	
1668 006260 000207			
1669			
1670			:ROUTINE TO SET CONSTANTS FOR WRITING/CHECKING 3 XOR PATTERN WITH
1671			:PARITY.
1672 006262 032702 000020	.XOR39: BIT	#20,R2	:CHECK BIT 3
1673 006266 001404	BEQ	.31S0	:BRANCH IF BIT 3 = 0
1674 006270 032702 002000	.31S1: BIT	#2000,R2	:CHECK BIT 9
1675 006274 001404	BEQ	.3NOT9	:BRANCH IF BIT 9 = 0
1676 006276 000407	BR	.31S9	
1677 006300 032702 002000	.31S0: BIT	#2000,R2	:CHECK BIT 9
1678 006304 001404	BEQ	.31S9	:BRANCH IF 0
1679 006306 005767 172442	.3NOT9: TST	ICOUNT	:CHECK IF NORMAL OR COMPLEMENT DATA
1680 006312 100004	BPL	LDCOMP	:GO LOAD COMPLEMENT CONSTANTS
1681 006314 100410	BMI	LDNORM	:GO LOAD NORMAL CONSTANTS
1682 006316 005767 172432	.31S9: TST	ICOUNT	:CHECK IF NORMAL OR COMPLEMENT DATA
1683 006322 100005	BPL	LDNORM	:GO LOAD NORMAL CONSTANTS
1684 006324 012700 177777	LDCOMP: MOV	#-1,R0	:SET COMPLEMENT CONSTANTS
1685 006330 012703 000401	MOV	#401,R3	
1686 006334 000207	RTS	PC	:RETURN
1687 006336 012700 000401	LDNORM: MOV	#401,R0	:LOAD NORMAL CONSTANTS
1688 006342 012703 177777	MOV	#-1,R3	
1689 006346 000207	RTS	PC	
1690			
1691			:ROUTINE TO CHECK 3 XOR 9 WORST CASE NOISE PATTERN
1692	:CALL: MOV	BANK#,-(SP)	:PUSH STARTING BANK # ONTO STACK
1693	: MOV	BLKCN,-(SP)	:AND 256. WORD BLOCK COUNT
1694	: JSR	PC,..3X9	:CALL ROUTINE
1695			
1696 006350 000240	..3X9: NOP		
1697 006352 004767 001104	JSR	PC,CKSWR	:GO CHECK SWITCH REGISTER
1698			
1699			:CHECK WORST CASE PATTERN
1700 006356 016604 000002	1\$: MOV	2(SP),R4	:GET 256. BLOCK WORD COUNT
1701 006362 016602 000004	MOV	4(SP),R2	:GET FIRST BANK #
1702 006366 004767 177366	JSR	PC,STMM2	:GO SET UP MEM MGMT
1703 006372 005000	CLR	R0	:SET CHECK WORD
1704 006374 005767 172354	TST	ICOUNT	:IF ICOUNT IS NEG AM CHECKING COMP-
1705 006400 000001	BPL	.+4	:LEMENTED PATTERN
1706 006402 005100	COM	R0	:SO COMPLEMENT CHECK WORD
1707 006404 012705 000040	2\$: MOV	#32.,R5	:SET 256. WORD COUNTER
1708			
1709 006410 005767 175106	3\$: TST	PARPAT	:BRANCH IF PARITY MEMORY PATTERN IS
1710 006414 001402	BEQ	30\$	:NOT TO BE CHECKED
1711			
1712 006416 004767 177640	JSR	PC,..XOR39	:GO GET CONSTANT
1713 006422			
1714 006422 012203	30\$: MOV	(R2)+,R3	:GET TEST DATA
1715 006424 020003	CMP	R0,R3	:COMPARE WITH CHECK WORD
1716 006426 001403	BEQ	.+10	
1717 006430 005046	CLR	-(SP)	:PUSH FAKE STATUS ON THE STACK
1718 006432 004767 172546	JSR	PC,ERROR	:ERROR! MEM DATA (R3) NOT = TEST DATA
1719			: (R0). ADDRESS=(R2)-2
1720			
1721 006436 012203	MOV	(R2)+,R3	:GET TEST DATA
1722 006440 020003	CMP	R0,R3	:COMPARE WITH CHECK WORD
1723 006442 001403	BEQ	.+10	

1724	006444	005046		CLR	-(SP)	; PUSH FAKE STATUS ON THE STACK
1725	006446	004767	172532	JSR	PC,ERROR	; ERROR! MEM DATA (R3) NOT = TEST DATA
1726						; (R0), ADDRESS=(R2)-2
1727						
1728	006452	012203		MOV	(R2)+,R3	; GET TEST DATA
1729	006454	020003		CMP	R0,R3	; COMPARE WITH CHECK WORD
1730	006456	001403		BEQ	.+10	
1731	006460	005046		CLR	-(SP)	; PUSH FAKE STATUS ON THE STACK
1732	006462	004767	172516	JSR	PC,ERROR	; ERROR! MEM DATA (R3) NOT = TEST DATA
1733						; (R0), ADDRESS=(R2)-2
1734						
1735	006466	012203		MOV	(R2)+,R3	; GET TEST DATA
1736	006470	020003		CMP	R0,R3	; COMPARE WITH CHECK WORD
1737	006472	001403		BEQ	.+10	
1738	006474	005046		CLR	-(SP)	; PUSH FAKE STATUS ON THE STACK
1739	006476	004767	172502	JSR	PC,ERROR	; ERROR! MEM DATA (R3) NOT = TEST DATA
1740						; (R0), ADDRESS=(R2)-2
1741						
1742	006502	012203		MOV	(R2)+,R3	; GET TEST DATA
1743	006504	020003		CMP	R0,R3	; COMPARE WITH CHECK WORD
1744	006506	001403		BEQ	.+10	
1745	006510	005046		CLR	-(SP)	; PUSH FAKE STATUS ON THE STACK
1746	006512	004767	172466	JSR	PC,ERROR	; ERROR! MEM DATA (R3) NOT = TEST DATA
1747						; (R0), ADDRESS=(R2)-2
1748						
1749	006516	012203		MOV	(R2)+,R3	; GET TEST DATA
1750	006520	020003		CMP	R0,R3	; COMPARE WITH CHECK WORD
1751	006522	001403		BEQ	.+10	
1752	006524	005046		CLR	-(SP)	; PUSH FAKE STATUS ON THE STACK
1753	006526	004767	172452	JSR	PC,ERROR	; ERROR! MEM DATA (R3) NOT = TEST DATA
1754						; (R0), ADDRESS=(R2)-2
1755						
1756	006532	012203		MOV	(R2)+,R3	; GET TEST DATA
1757	006534	020003		CMP	R0,R3	; COMPARE WITH CHECK WORD
1758	006536	001403		BEQ	.+10	
1759	006540	005046		CLR	-(SP)	; PUSH FAKE STATUS ON THE STACK
1760	006542	004767	172436	JSR	PC,ERROR	; ERROR! MEM DATA (R3) NOT = TEST DATA
1761						; (R0), ADDRESS=(R2)-2
1762						
1763	006546	012203		MOV	(R2)+,R3	; GET TEST DATA
1764	006550	020003		CMP	R0,R3	; COMPARE WITH CHECK WORD
1765	006552	001403		BEQ	.+10	
1766	006554	005046		CLR	-(SP)	; PUSH FAKE STATUS ON THE STACK
1767	006556	004767	172422	JSR	PC,ERROR	; ERROR! MEM DATA (R3) NOT = TEST DATA
1768						; (R0), ADDRESS=(R2)-2
1769						
1770						
1771	006562	005100		COM	R0	; COMPLEMENT CHECK WORD
1772	006564	005305		DEC	R5	; DECREMENT 256. WORD COUNTER
1773	006566	001310		BNE	3\$	
1774	006570	005100		COM	R0	; COMPLEMENT CHECK WORD
1775	006572	005304		DEC	R4	; DECREMENT BLOCK COUNTER
1776	006574	001303		BNE	2\$	
1777						
1778	006576	032737	040000 177570	BIT	#40000,@#SWR	; LOOP ON TEST?
1779	006604	001264		BNE	1\$	; BRANCH IF LOOP ON TEST DESIRED

1780	006606	016667	000002	172152	40\$:	MOV	2(SP),COUNT	;GET # OF 256. WORD BLOCKS TO CHECK
1781	006614	016602	000004			MOV	4(SP),R2	;GET STARTING BANK #
1782	006620	004767	177134			JSR	PC,STMM2	;GO SET UP MEM MGMT IF REQUIRED
1783								
1784							;CHECK WORST CASE BIT COMPLEMENT PATTERN	
1785	006624	005000				CLR	R0	
1786	006626	005767	172122			TST	ICOUNT	;CHECK IF COMPLEMENT PATTERN
1787	006632	100001				BPL	.+4	
1788	006634	005100				COM	R0	;COMPLEMENT CHECK WORD
1789	006636	012704	000040		4\$:	MOV	#32.,R4	;SET 256. WORD COUNTER
1790	006642	012705	000010		5\$:	MOV	#8.,R5	;SET 8 WORD COUNTER
1791	006646	005767	174650		6\$:	TST	PARPAT	;BRANCH IF PARITY MEMORY PATTERN IS
1792	006652	001402				BEQ	60\$	;NOT TO BE CHECKED
1793	006654	004767	177402			JSR	PC,XOR39	
1794	006660	012203			60\$:	MOV	(R2)+,R3	;GET DATA
1795	006662	020003				CMP	R0,R3	;CHECK DATA
1796	006664	001403				BEQ	.+10	
1797	006666	005046				CLR	-(SP)	
1798	006670	004767	172310			JSR	PC,ERROR	
1799	006674	005100				COM	R0	;COMPLEMENT CHECK WORD
1800	006676	005142				COM	-(R2)	;COMPLEMENT TEST DATA
1801	006700	012203				MOV	(R2)+,R3	;GET DATA
1802	006702	020003				CMP	R0,R3	;CHECK
1803	006704	001403				BEQ	.+10	
1804	006706	005046				CLR	-(SP)	;PUSH FAKE STATUS ON THE STACK
1805	006710	004767	172270			JSR	PC,ERROR	
1806	006714	005100				COM	R0	;COMPLEMENT CHECK WORD
1807	006716	005162	177776			COM	-2(R2)	;RESTORE DATA
1808	006722	005305				DEC	R5	;DECREMENT 4 WORD COUNTER
1809	006724	001350				BNE	6\$	
1810	006726	005100				COM	R0	;COMPLEMENT CHECK WORD
1811	006730	005304				DEC	R4	;DECREMENT 256. WORD COUNTER
1812	006732	001343				BNE	5\$	
1813	006734	005100				COM	R0	;COMPLEMENT CHECK WORD
1814	006736	005367	172024			DEC	COUNT	;DECREMENT BLOCK COUNTER
1815	006742	001335				BNE	4\$	
1816								
1817	006744	016602	000004			MOV	4(SP),R2	;GET BANK #
1818	006750	004767	177004			JSR	PC,STMM2	
1819	006754	016603	000002			MOV	2(SP),R3	;GET BLOCK COUNT
1820	006760	032737	040000	177570		BIT	#40000,@#SWR	;LOOP ON TEST
1821	006766	001307				BNE	40\$	;BRANCH IF LOOP ON TEST
1822	006770	006367	171760			ASL	ICOUNT	
1823	006774	102402				BVS	7\$	
1824	006776	000167	177354			JMP	1\$	
1825	007002	012705	000020		7\$:	MOV	#16.,R5	;COMPLEMENT PATTERN
1826	007006	011200			10\$:	MOV	(R2),R0	;GET 1ST DATA WORD
1827	007010	016204	000020			MOV	20(R2),R4	;GET 9TH DATA WORD
1828	007014	110422				MOVB	R4,(R2)+	;SWAP WORDS 1-8
1829	007016	110422				MOVB	R4,(R2)+	;WITH 9-16
1830	007020	110422				MOVB	R4,(R2)+	
1831	007022	110422				MOVB	R4,(R2)+	
1832	007024	110422				MOVB	R4,(R2)+	
1833	007026	110422				MOVB	R4,(R2)+	
1834	007030	110422				MOVB	R4,(R2)+	
1835	007032	110422				MOVB	R4,(R2)+	

1836	007034	110422	MOVB	R4,(R2)+	
1837	007036	110422	MOVB	R4,(R2)+	
1838	007040	110422	MOVB	R4,(R2)+	
1839	007042	110422	MOVB	R4,(R2)+	
1840	007044	110422	MOVB	R4,(R2)+	
1841	007046	110422	MOVB	R4,(R2)+	
1842	007050	110422	MOVB	R4,(R2)+	
1843	007052	110422	MOVB	R4,(R2)+	
1844	007054	110022	MOVB	R0,(R2)+	;AND VICE VERSA
1845	007056	110022	MOVB	R0,(R2)+	
1846	007060	110022	MOVB	R0,(R2)+	
1847	007062	110022	MOVB	R0,(R2)+	
1848	007064	110022	MOVB	R0,(R2)+	
1849	007066	110022	MOVB	R0,(R2)+	
1850	007070	110022	MOVB	R0,(R2)+	
1851	007072	110022	MOVB	R0,(R2)+	
1852	007074	110022	MOVB	R0,(R2)+	
1853	007076	110022	MOVB	R0,(R2)+	
1854	007100	110022	MOVB	R0,(R2)+	
1855	007102	110022	MOVB	R0,(R2)+	
1856	007104	110022	MOVB	R0,(R2)+	
1857	007106	110022	MOVB	R0,(R2)+	
1858	007110	110022	MOVB	R0,(R2)+	
1859	007112	110022	MOVB	R0,(R2)+	
1860	007114	005305	DEC	R5	
1861	007116	001333	BNE	10\$	
1862	007120	005303	DEC	R3	
1863	007122	001327	BNE	7\$	
1864					
1865	007124	005767	TST	ICOUNT	
1866	007130	001402	BEQ	11\$	
1867	007132	000167	JMP	1\$	
1868	007136	012616	11\$: MOV	(SP)+,(SP)	
1869	007140	012616	MOV	(SP)+,(SP)	
1870	007142	000207	RTS	PC	
1871					
1872					
1873					
1874					
1875					
1876					
1877					
1878	007144	012616	.8x13: MOV	(SP)+,(SP)	;ADJUST STACK
1879	007146	012616	MOV	(SP)+,(SP)	
1880	007150	000207	RTS	PC	
1881					
1882					
1883					
1884					
1885					
1886					
1887					
1888	007152	012616	..8x13: MOV	(SP)+,(SP)	
1889	007154	012616	MOV	(SP)+,(SP)	
1890	007156	000207	RTS	PC	;RETURN
1891					

;ROUTINE TO WRITE 8 XOR 13 WORST CASE NOISE TEST PATTERN
.SBTTL 8 XOR 13 ROUTINES
;CALL: MOV BANK #,-(SP)
; MOV #4KBANKS,-(SP)
; JSR PC,.8x13
.8x13: MOV (SP)+,(SP) ;ADJUST STACK
MOV (SP)+,(SP)
RTS PC
;ROUTINE TO CHECK 8 XOR 13 WORST CASE NOISE TEST PATTERN
;CALL:
; MOV BANK #,-(SP) ;PUSH FIRST BANK # ON THE STACK
; MOV #BANKS,-(SP) ;PUSH # OF 4K BANKS TO CHECK ON THE STACK
; JSR PC,..8x13 ;CALL ROUTINE
..8x13: MOV (SP)+,(SP)
MOV (SP)+,(SP)
RTS PC ;RETURN

Address	Hex	Hex	Hex	Assembly	Comments
1892				.SBTTL ROTATING 1'S & 0'S ROUTINES	
1893				:ROUTINE TO CHECK ROTATING '0' BIT THROUGH FIELD OF 1'S	
1894				:CALL: MOV BANK#,-(SP)	:SET STARTING BANK #
1895				: MOV BLKCNT,-(SP)	:SET 256. WORD BLOCK COUNT
1896				: JSR PC,.ROTO	:CALL ROUTINE
1897					
1898	007160	004767	000276	.ROTO: JSR PC,CKSWR	:GO CHECK SWITCHES
1899	007164	016604	000002	MOV 2(SP),R4	:GET 256. WORD BLOCK COUNT
1900	007170	016602	000004	MOV 4(SP),R2	:GET FIRST BANK #
1901	007174	004767	176560	JSR PC,STMM2	:GO SET UP MEM MGMT (IF AVAIL)
1902	007200	012700	177777	MOV #-1,R0	:SET CHECK WORD
1903					
1904	007204	012705	000400	1\$: MOV #256.,R5	:SET 256. WORD COUNT
1905	007210	000241		2\$: CLC	:CLEAR CARRY BIT IN PSW
1906	007212	004767	000124	JSR PC,ROTATE	
1907	007216	016203	177776	MOV -2(R2),R3	:GET RESULT
1908	007222	103402		BCS 3\$	:BRANCH IF 'C' BIT WAS SET
1909	007224	020003		CMP R0,R3	:CHECK RESULT
1910	007226	001403		BEQ 4\$	
1911	007230	005046		3\$: CLR -(SP)	:ERROR! COULD NOT ROTATE '0' BIT
1912	007232	004767	171746	JSR PC,ERROR	:THROUGH ADDRESS IN R2
1913	007236	005305		4\$: DEC R5	:DECREMENT 256. WORD COUNT
1914	007240	001363		BNE 2\$	:LOOP UNTIL DONE
1915	007242	005304		DEC R4	:DECREMENT 256. WORD BLOCK COUNT
1916	007244	001357		BNE 1\$	:LOOP UNTIL DONE
1917	007246	012616		MOV (SP)+,(SP)	:POP CONSTANTS OFF THE STACK
1918	007250	012616		MOV (SP)+,(SP)	
1919	007252	000207		RTS PC	:RETURN TO CALLER
1920					
1921				:ROUTINE TO CHECK ROTATING '1' BIT THROUGH A FIELD OF 0'S	
1922				:CALL: MOV BANK#,-(SP)	:SET STARTING BANK #
1923				: MOV BLKCNT,-(SP)	:SET # OF 256. WORD BLOCKS TO CHECK
1924				: JSR PC,.ROT1	:CALL ROUTINE
1925					
1926	007254	004767	000202	.ROT1: JSR PC,CKSWR	:GO CHECK SWITCHES
1927	007260	016604	000002	MOV 2(SP),R4	:GET # OF 256. WORD BLOCKS TO CHECK
1928	007264	016602	000004	MOV 4(SP),R2	:GET STARTING BANK #
1929	007270	004767	176464	JSR PC,STMM2	:GO SET UP MEM MGMT (IF AVAIL)
1930	007274	005000		CLR R0	:SET CHECK WORD
1931					
1932	007276	012705	000400	1\$: MOV #256.,R5	:SET 256. WORD COUNTER
1933	007302	000261		2\$: SEC	:SET 'C' BIT IN PSW
1934	007304	004767	000032	JSR PC,ROTATE	:GO ROTATE '1' BIT
1935	007310	016203	177776	MOV -2(R2),R3	:GET RESULT
1936	007314	103002		BCC 3\$	:BRANCH IF 'C' IS CLEAR
1937	007316	020003		CMP R0,R3	:CHECK RESULT
1938	007320	001401		BEQ .+4	
1939	007322	104400		3\$: HLT	:ERROR! COULD NOT ROTATE '1' BIT
1940					:THROUGH ADDRESS IN R2
1941	007324	005305		DEC R5	:DECREMENT 256. WORD COUNT
1942	007326	001365		BNE 2\$	
1943	007330	005304		DEC R4	:DECREMENT 256. WORD BLOCK COUNT
1944	007332	001361		BNE 1\$	
1945	007334	012616		MOV (SP)+,(SP)	:ADJUST RETURN ADDRESS
1946	007336	012616		MOV (SP)+,(SP)	
1947	007340	000207		RTS PC	:RETURN TO CALLER

```

1948
1949
1950 007342 106112
1951 007344 106112
1952 007346 106112
1953 007350 106112
1954 007352 106112
1955 007354 106112
1956 007356 106112
1957 007360 106112
1958 007362 106122
1959 007364 106112
1960 007366 106112
1961 007370 106112
1962 007372 106112
1963 007374 106112
1964 007376 106112
1965 007400 106112
1966 007402 106112
1967 007404 106122
1968 007406 000207
1969
1970
1971
1972
1973
1974
1975 007410 016604 000002
1976 007414 016602 000004
1977 007420 004767 176334
1978 007424 012700
1979 007426 000000
1980 007430 012703 000100
1981 007434 010022
1982 007436 010022
1983 007440 010022
1984 007442 010022
1985 007444 005303
1986 007446 001372
1987 007450 005304
1988 007452 001366
1989 007454 012616
1990 007456 012616
1991 007460 000207
1992
1993
1994
1995
1996
1997 007462 042767 017777 171270
1998 007470 032737 000400 177570
1999 007476 001402
2000 007500 004767 000464
2001 007504 032737 001000 177570
2002 007512 001404
2003 007514 056767 171236 171236

;ROUTINE TO ROTATE 'C' BIT THROUGH A MEMORY LOCATION.
ROTATE: ROLB (R2) ;(R2)=177776 OR 000001
        ROLB (R2) ;(R2)=177775 OR 000002
        ROLB (R2) ;(R2)=177773 OR 000004
        ROLB (R2) ;(R2)=177767 OR 000010
        ROLB (R2) ;(R2)=177757 OR 000020
        ROLB (R2) ;(R2)=177737 OR 000040
        ROLB (R2) ;(R2)=177677 OR 000100
        ROLB (R2) ;(R2)=177777 OR 000000
        ROLB (R2)+ ;(R2)=177577 OR 000200
        ROLB (R2)+ ;(R2)=177377 OR 000400
        ROLB (R2)+ ;(R2)=176777 OR 001000
        ROLB (R2)+ ;(R2)=175777 OR 002000
        ROLB (R2)+ ;(R2)=173777 OR 004000
        ROLB (R2)+ ;(R2)=167777 OR 010000
        ROLB (R2)+ ;(R2)=157777 OR 020000
        ROLB (R2)+ ;(R2)=137777 OR 040000
        ROLB (R2)+ ;(R2)=077777 OR 100000
        ROLB (R2)+ ;(R2)=177777 OR 000000
        RTS PC ;RETURN

;ROUTINE TO WRITE ONE WORD PATTERN INTO MEMORY
;CALL: MOV BANK#,-(SP) ;PUSH STARTING BANK # ONTO STACK
;      MOV BLKCNT,-(SP) ;AND 128. WORD BLOCK COUNT
;      JSR PC,WRTPAT ;CALL ROUTINE
WRTPAT: MOV 2(SP),R4 ;GET BLOCK COUNT
        MOV 4(SP),R2 ;GET STARTING BANK #
        JSR PC,STMM2 ;GO SET UP MEM MGMT
        MOV (PC)+,R0 ;GET USER CONSTANT
;CONST: 0
1$: MOV #64.,R3 ;SET 256. WORD COUNTER
2$: MOV R0,(R2)+ ;WRITE 256. WORDS
    MOV R0,(R2)+
    MOV R0,(R2)+
    MOV R0,(R2)+
    DEC R3 ;DECREMENT 256. WORD COUNTER
    BNE 2$ ;LOOP UNTIL 256. WORDS HAVE BEEN WRITTEN
    DEC R4 ;DECREMENT BLOCK COUNT
    BNE 1$
    MOV (SP)+,(SP) ;ADJUST STACK
    MOV (SP)+,(SP)
    RTS PC

;ROUTINE TO CHECK THE SWITCH REGISTER
;CHECK SWITCH 9: IF SET, LOAD ERROR COUNT INTO THE DISPLAY REGISTER;
;IF NOT SET, LOAD PASS COUNT INTO THE DISPLAY REGISTER
CKSWR: BIC #17777,LDDISP ;SAVE RELOCATION BITS
        BIT #BIT8,@NSWR ;CHECK SWITCH 8
        BEQ 10$ ;BRANCH IF SET
        JSR PC,REL24K ;GO RELOCATE PROGRAM BACK TO 4K AND STOP
10$: BIT #BIT9,@NSWR ;SWITCH 9 SET ?
        BEQ 1$
        BIS ERcnt,LDDISP ;LOAD ERROR COUNT
    
```


2004	007522	000403			BR	2\$	
2005	007524	056767	171222	171226	1\$: BIS	ICNT,LDDISP	:LOAD PASS COUNT
2006	007532	016737	171222	177570	2\$: MOV	LDDISP,@#DISPLAY	:LOAD THE DISPLAY REGISTER
2007	007540	012767	040177	171206	MOV	#040177,ICOUNT	:LOAD ITERATION COUNT WORD
2008	007546	032737	004000	177570	BIT	#4000,@#SWR	:CHECK SW11
2009	007554	001402			BEQ	.+6	
2010	007556	105067	171172		CLRB	ICOUNT	:ICOUNT =040000 IF SW11 =1
2011	007562	000207			RTS	PC	
2012							
2013							:MESSAGES
2014	007564	005015	047524	051040	RESLDR:	.ASCIZ	<15><12>'TO RESTORE LOADERS START AT 162'<15><12>
2015	007572	051505	047524	042522			
2016	007600	046040	040517	042504			
2017	007606	051522	051440	040524			
2018	007614	052122	040440	020124			
2019	007622	033061	006462	000012			
2020	007630	005015	047105	041101	PARITY:	.ASCIZ	<15><12>'ENABLE PARITY? 1/0=YES/NO '
2021	007636	042514	050040	051101			
2022	007644	052111	037531	030440			
2023	007652	030057	054475	051505			
2024	007660	047057	020117	000			
2025	007665	015	051412	040524	STBANK:	.ASCIZ	<15><12>'STARTING BANK #(8)? '
2026	007672	052122	047111	020107			
2027	007700	040502	045516	021440			
2028	007706	034050	037451	000040			
2029	007714	005015	020043	043117	BANKS:	.ASCIZ	<15><12>'# OF 4K BANKS TO TEST(8)? '
2030	007722	032040	020113	040502			
2031	007730	045516	020123	047524			
2032	007736	052040	051505	024124			
2033	007744	024470	020077	000			
2034	007751	015	050012	052101	PAT:	.ASCIZ	<15><12>'PATTERN #? '
2035	007756	042524	047122	021440			
2036	007764	020077	000				
2037	007767	015	037412	000	QUEST:	.ASCIZ	<15><12>'?'
2038	007773	015	052012	050131	CONST:	.ASCIZ	<15><12>'TYPE CONSTANT'
2039	010000	020105	047503	051516			
2040	010006	040524	052116	000			
2041	010013	015	044412	050116	PRG3M:	.ASCIZ	<15><12>'INPUT # OF 256. WORD BLOCKS TO TEST INSTEAD OF '
2042	010020	052125	021440	047440			
2043	010026	020106	032462	027066			
2044	010034	053440	051117	020104			
2045	010042	046102	041517	051513			
2046	010050	052040	020117	042524			
2047	010056	052123	044440	051516			
2048	010064	042524	042101	047440			
2049	010072	000106					
2050	010074	005015	054524	042520	PRG4M:	.ASCIZ	<15><12>'TYPE ADDRESS'
2051	010102	040440	042104	042522			
2052	010110	051523	000				
2053	010113	015	052012	020117	RELOCM:	.ASCIZ	<15><12>'TO RESTORE PROGRAM START AT '
2054	010120	042522	052123	051117			
2055	010126	020105	051120	043517			
2056	010134	040522	020115	052123			
2057	010142	051101	020124	052101			
2058	010150	000040					
2059	010152	000052			ASTERISK:	.ASCIZ	'*'

```

2060 010154 042503 045115 020101 ENDMMSG: .ASCIZ 'CEMJA DONE!'
2061 010162 047504 042516 000041
2062                                     .EVEN
2063
2064 ;ROUTINE TO RELOCATE PROGRAM BACK TO 0
REL24K: MOV     PC,R0                ;FORM BASE ADDRESS WHERE CODE
        BIC     #17777,R0           ;IS RELOCATED
        MOV     R0,1$              ;PUT FROM ADDRESS INTO SUBROUTINE CALL
        JSR     R5,RELOC           ;RELOCATE CODE TO
                                     ;LOWEST 4K
1$:      0
        0
        MOV     #STKPTR,SP          ;SET STACK PTR
        BIC     #100000,@#LDDISP    ;CLEAR RELOCATION INDICATOR
        MOV     @#LDDISP,@#DISPLAY  ;LOAD DISPLAY REGISTER
        CLR     @#RELOC             ;CLEAR RELOCATION FACTOR
        RESET
        JMP     @#PONE              ;DISABLE MEM MGMT
                                     ;RESTORE LOADERS & HALT
;
;CLRPAR: CLR     -(R2)              ;CLR MEM OF PAR ERR
        JMP     @PERSTRT            ;RESTART SELETED PROGRAM
;
;PARCLR: CLR     -(R2)              ;CLR MEM OF PAR ERR
        RTI                          ;CONTINUE SCAN
;
;LODAR.
        .END
2085 010256 000001

```


STMM2	005760	1586#	1626	1702	1782	1818	1901	1929	1977					
SWR	= 177570	397#	646	671	676	680	708	712	922	927	1418	1425	1499	1548
		1778	1820	1998	2001	2008								
T	= 000020	360#												
TBITVE	= 000014	374#												
TkB	= 177562	394#												
TkS	= 177560	393#	523	564*										
TkVEC	= 000060	381#												
TPB	= 177566	396#												
TPS	= 177564	395#	1268											
TPVEC	= 000064	382#												
TRAPVE	= 000034	380#	466											
TRAP4	001460	656	719#											
TRTVEC	= 000014	375#												
TYPIT	001110	628	636	639#										
UBREAK	= 177770	391#												
UM	= 140000	365#												
UP	= 000000	443#	1200	1553	1554	1557	1602	1603						
V	= 000002	357#	1582											
WDS.25	003602	1152#	1234	1238	1245	1249								
WRTPAT	007410	1235	1246	1975#										
WRTUP	002616	932	942#											
XMTDAT	001501	698	723#											
Z	= 000004	358#												
\$CRLF	000746	476	578#											
\$ENDAD	004304	470	1117	1273#										
\$FILL	001021	612#												
\$FORMO	001554	654	684	690	734#	953								
\$LDR	002112	823#	904											
\$LDRM	002264	858	864#											
\$LDR1	002262	834*	848	862#										
\$NULL	001020	611#	631											
\$PRINT	000770	475	570	597#	655	657	685	687	691	697	702	710	805	857
		905	954	1262	1266	1330	1353	1385	1428	1431	1436	1450	1454	1458
		1465	1467											
\$RESTR	000300	507#	563	707	779	816								
\$RLDR	002210	482	845#	1272										
\$SAVR	000254	495#	524	682	736	797								
\$TKFLG	001023	615#												
\$TPB	001026	617#	641*											
\$TPFLG	001022	613#												
\$TPS	001024	616#	639											
\$\$\$TAG	= 016146	2085#												
.	= 010256	445#	460#	466#	469#	472#	474#	480#	485#	488#	518#	582#	640	647
		677	713	732#	756	829	868#	886#	1060	1108	1269	1305	1405#	1494#
		1705	1716	1723	1730	1737	1744	1751	1758	1765	1787	1796	1803	1938
		2009	2085											
.CONST	007426	1232*	1243*	1979#										
.MAMF	005410	561	937	1126	1376	1499#								
.PARSR	004540	1346#	1511											
.ROTO	007160	1239	1898#											
.ROT1	007254	1250	1926#											
.TYPE	001030	601	618#											
.XOR39	006262	1663	1672#	1712	1793									
.3x9	006350	1157	1696#											
.8x13	007152	1186	1888#											

.22PAR	005020	1413#	1516	
.31S0	006300	1673	1677#	
.31S1	006270	1674#		
.31S9	006316	1676	1678	1682#
.3N019	006306	1675	1679#	
.3X9	006130	1145	1625#	
.8X13	007144	1175	1878#	

COMMEN	1#
ENDCOM	1#
ESCAPE	1#
GETPRI	1#
GETSWR	1#
MULT	1#
NEWTST	1#
POP	1#
PUSH	1#
REPORT	1#
SETPRI	1#
SETUP	1#
SKIP	1#
SLASH	1#
STARS	1#
SWRSU	1#
TYPBIN	1#
TYPDEC	1#
TYPNAM	1#
TYPNUM	1#
TYPOCS	1#
TYPOCT	1#
TYPTXT	1#
\$TYPE	322#
\$ESCA	1#
\$NEWT	1#
\$SKIP	1#
.EQUAT	1#
.HEADE	1#
.KT11	1#
.SETUP	1#
.SWRHI	1#
.\$ACT1	1#
.\$APT8	1#
.\$APTH	1#
.\$APTY	1#
.\$ASTA	1#
.\$CATC	1#
.\$CMTA	1#
.\$DB2D	1#
.\$DB20	1#
.\$DIV	1#
.\$EOP	1#
.\$ERRO	1#
.\$ERRT	1#
.\$MULT	1#
.\$POWE	1#
.\$RAND	1#
.\$RDDE	1#
.\$RDOC	1#
.\$READ	1#
.\$R2AZ	1#
.\$SAVE	1#
.\$SB2D	1#
.\$SB20	1#
.\$SCOP	1#

.SSIZE 1#
.SSUPR 1#
.STRAP 1#
.STYPB 1#
.STYPD 1#
.STYPE 1#
.STYPO 1#
.S4OCA 1#
.1170 1#

. ABS. 010256 000

ERRORS DETECTED: 0

CEMJAD.BIN,CEMJAD.LST/CRF/SOL/NL:TOC=CEMJAD.SML,CEMJAD.P11
RUN-TIME: 8 10 .4 SECONDS
RUN-TIME RATIO: 98/19=5.0
CORE USED: 32K (63 PAGES)