

M11/ME11

07124K MEMORY EXERCISER 16K VERIFY
MD-11-DZQMC-B

EP-DZQMC-B-DL-A

OCT 1976

COPYRIGHT ©1976

digital

FICHE 1 OF 1

Made in U.S.A.

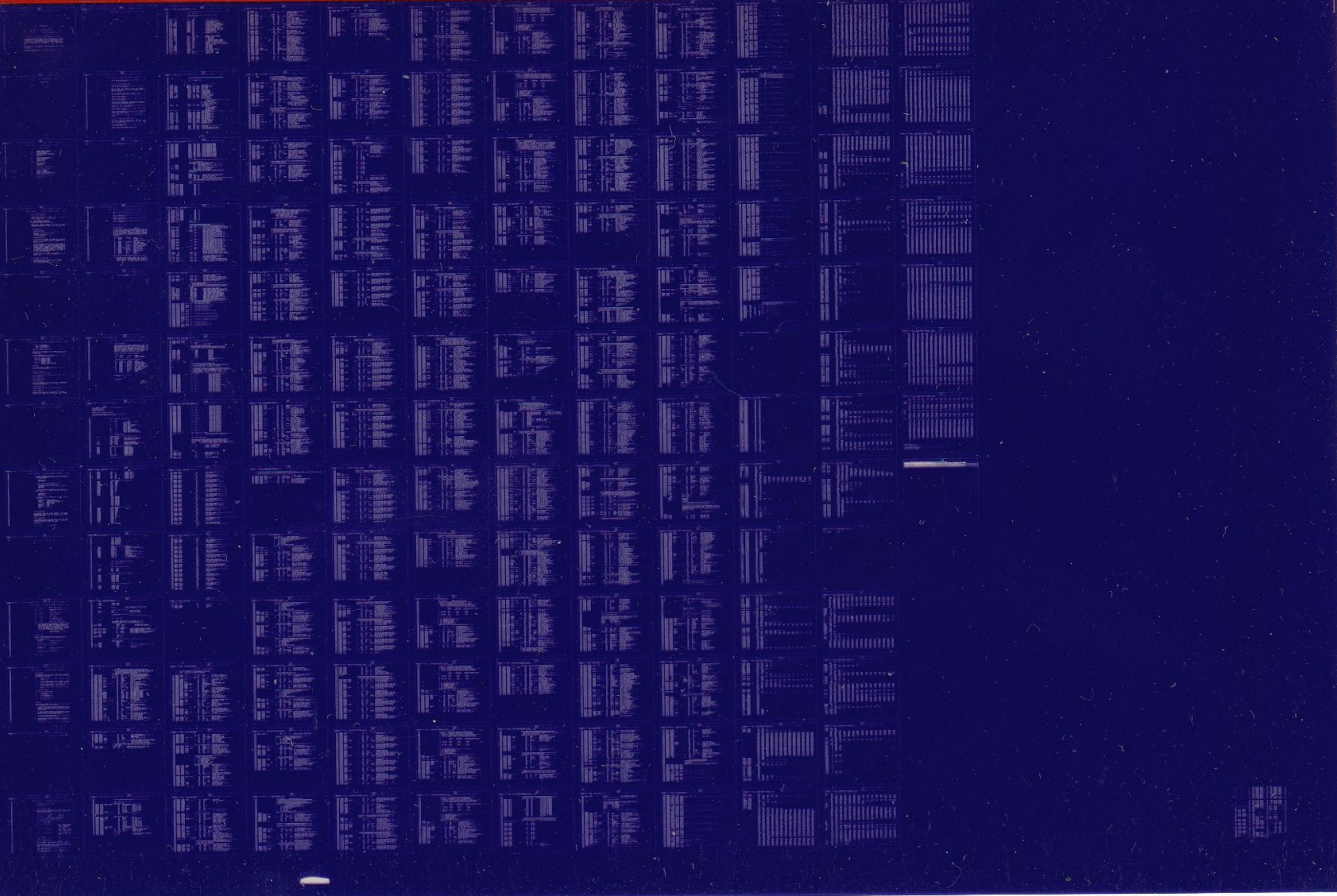


TABLE OF CONTENTS

1.0	GENERAL PROGRAM INFORMATION.
1.1	PROGRAM PURPOSE (ABSTRACT)
1.2	SYSTEM REQUIREMENTS
1.3	RELATED DOCUMENTS AND STANDARDS
1.4	DIAGNOSTIC HIERARCHY PREREQUISITES
1.5	ASSUMPTIONS
2.0	OPERATING INSTRUCTIONS
2.1	LOADING AND STARTING PROCEDURE
2.2	SPECIAL ENVIRONMENTS
2.3	PROGRAM OPTIONS
2.4	EXECUTION TIMES
3.0	ERROR INFORMATION
3.1	ERROR REPORTING
3.2	ERROR HALTS
4.0	PERFORMANCE AND PROGRESS REPORTS
5.0	DEVICE INFORMATION TABLES
6.0	SUB-TEST SUMMARIES
6.1	SECTION 1: ADDRESS TESTS
6.2	SECTION 2: WORST CASE NOISE TESTS
6.3	SECTION 3: INSTRUCTION EXECUTION TESTS
6.4	SPECIAL TOGGLE IN TESTS
7.0	PROGRAM FUNCTIONAL FLOW CHARTS
8.0	PROGRAM LISTING

001
002
003
004
005
006
007
008
009
010
011
012
013
014
015
016
017
018
019
020
021
022
023
024
025
026
027
028
029
030
031
032
033
034
035
036
037
038
039
040
041
042
043
044
045
046
047
048
049
050
051
052
053
054
055
056
057
058
059
060
061
062
063
064
065
066
067
068
069
070
071
072
073
074
075
076
077
078
079
080
081
082
083
084
085
086
087
088
089
090
091
092
093
094
095
096
097
098
099
100

F01

MAINDEC-11-DZQMC-B-D:
DZQMCB.P11

0-124K MEMORY EXERCISER, 16K VER

MACY11 27(732) 10-SEP-76 12:01 PAGE 6

144
145

DOCUMENT NO. DEC-11-HMFMA-B-D
E. APPLICABLE CIRCUIT SCHEMATICS:

G235 - 16K X-Y DRIVE
G114 - 16K SENSE/INHIBIT
M8293 - 16K UNIBUS TIMING
M7259 - PARITY CONTROL

1.4 DIAGNOSTIC HIERARCHY PREREQUISITES

BEFORE RUNNING THIS PROGRAM, A CPU DIAGNOSTIC SHOULD BE RUN TO VERIFY THE FUNCTIONALITY OF THE PROCESSOR AND PDP-11 INSTRUCTION SET.

IF MEMORY MANAGEMENT IS TO BE USED, THEN THE KT11 DIAGNOSTIC SHOULD ALSO BE RUN BEFORE THIS PROGRAM.

PDP-11/05 - MAINDEC-11-DZQKC
PDP-11/20 - MAINDEC-11-DZQKC
PDP-11/40 - MAINDEC-11-DBQEA
OR MAINDEC-11-DCQKC
PDP-11/45 - MAINDEC-11-DCQKC
KT11-C - MAINDEC-11-DCKTA THRU DCKTF
KT11-D - MAINDEC-11-DBKTA THRU DBKTF

1.5 ASSUMPTIONS

THIS PROGRAM ASSUMES THE CORRECT OPERATION OF THE CPU AND, IF USED, THE MEMORY MANAGEMENT OPTION.

2.0 OPERATING INSTRUCTIONS

2.1 LOADING AND STARTING PROCEDURES

2.1.1 LOAD THE PROGRAM USING ANY STANDARD ABSOLUTE LOADER.

2.1.2 STARTING ADDRESS 200:

NORMAL PROGRAM EXECUTION.

2.1.3 STARTING ADDRESS 204:

ALLOWS THE OPERATOR TO INPUT, VIA TELETYPE CONVERSATION, FIRST AND LAST ADDRESSES TO BE EXERCISED, AND A DATA PATTERN TO BE USED IN TESTS 6 AND 7.

2.1.4 STARTING ADDRESS 210:

RESTART PROGRAM USING PREVIOUSLY SELECTED PARAMETERS.

2.1.5 STARTING ADDRESS 214:

RESTORE LOADERS AND HALT. THIS ROUTINE IS CAPABLE OF RELOCATING THE PROGRAM BACK TO BANKS 0 AND 1 IF THE PROGRAM WAS HALTED WHILE RUNNING THE TOP TWO BANKS OF MEMORY. THERE

146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201

H01

MAINDEC-11-DZQMC-B-D: 0-124K MEMORY EXERCISER, 16K VER
DZQMCB.P11

MACY11 27(732) 10-SEP-76 12:01 PAGE 8

END

ARE SPECIAL PROCEDURES REQUIRED FOR THIS SITUATION.

2

203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258

A. IF MEMORY ADDRESSES 0-1000 HAVE NOT BEEN EXERCISED, EITHER THROUGH PARAMETER SELECTION (SA=204) OR BY RUNNING WITH SW05=1, THEN:

LOAD ADDRESS 214.
PRESS START.

B. IF RUNNING WITHOUT MEMORY MANAGEMENT, THEN:

LOAD ADDRESS <214+RELOCATION FACTOR>
(RELOCATION FACTOR IS TYPED WHEN THE PROGRAM IS RELOCATED),
PRESS START.

C. IF RUNNING WITH MEMORY MANAGEMENT AND THE UNIBUS HAS NOT BEEN INITIALIZED (VIA RESET INSTRUCTION, START SWITCH, ETC.), THEN:

LOAD ADDRESS 777707 (PC)
DEPOSIT 214
PRESS CONTINUE

D. IF RUNNING WITH MEMORY MANAGEMENT AND THE UNIBUS HAS BEEN INITIALIZED:

LOAD ADDRESS 772340 (KIPARC)
DEPOSIT <((RELOCATION FACTOR)/100)>
(EXAMPLE: RELOCATION FACTOR=540000, THEN DEPOSIT 005400)
LOAD ADDRESS 777572 (SRO)
DEPOSIT 000001
LOAD ADDRESS 777707 (PC)
DEPOSIT 214
PRESS CONTINUE

2.1.6 STARTING ADDRESS 220:

BYTE ADDRESS MEMORY MAP TYPEOUT ROUTINE. THIS ROUTINE PERFORMS DATI, DATIP, DATO, AND DATOB ON ALL POSSIBLE ADDRESSES, AND TYPES THE RANGES OF ADDRESSES WHICH DO NOT CAUSE A TIMEOUT TRAP.

2.2 SPECIAL ENVIRONMENTS

IF THE PROGRAM IS RUN IN QUICK VERIFY MODE UNDER ACT11 THE PROGRAM IS DONE AFTER THE FIRST PASS. ALSO, THE PROGRAM DOES NOT RELOCATE TO TEST THE LOWER 8K OF MEMORY.

2.3 PROGRAM OPTIONS

SW!5 = 1 OR UP.... HALT ON ERROR

4r

J01

MAINDEC-11-DZQMC-B-D: 0-124K MEMORY EXERCISER. 16K VER
DZQMCB.P11

MACY11 27(732) 10-SEP-76 12:01 PAGE 10

359
360

SW14 = 1 OR UP....

LOOP ON TEST

251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
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

- SW13 = 1 OR UP.... INHIBIT ERROR TYPEOUT
- SW12 = 1 OR UP.... INHIBIT MEMORY MANAGEMENT (INITIAL START ONLY)
- SW11 = 1 OR UP.... INHIBIT SUBTEST ITERATION
- SW10 = 1 OR UP.... RING BELL ON ERROR
- SW9 = 1 OR UP.... LOOP ON ERROR
- SW8 = 1 OR UP.... LOOP ON TEST IN SWR<4:0>
- SW7 = 1 OR UP.... INHIBIT PROGRAM RELOCATION
- SW6 = 1 OR UP.... INHIBIT PARITY ERROR DETECTION

NOTE: WITH PARITY ERROR DETECTION ENABLED, A MEMORY FAILURE WHILE RUNNING THE WORSE CASE NOISE TESTS (NON-PARITY) CAN CAUSE A PARITY ERROR. THE ERROR PRINTOUT ON A PARITY ERROR DOES NOT TYPE THE GOOD DATA. THUS A BIT DROP OR PICKUP WILL NOT BE TYPED AS SUCH. IT IS BEST TO RUN THE PROGRAM FOR 1 PASS WITH PARITY DISABLED, THEN, RESTART THE PROGRAM WITH PARITY ENABLED.

- SW5 = 1 OR UP.... INHIBIT EXERCISING VECTOR AREA (LOCATIONS 0-1000).

2.4 EXECUTION TIMES

EXECUTION TIME IS DEPENDENT ON TYPE OF MEMORY, AND AMOUNT OF MEMORY. WORSE CASE RUN TIMES WITH 900NS MEMORYS ARE:

- A. FOR NON-PARITY MEMORY
 - FIRST PASS: 45 SECONDS + 3 SECONDS PER 4K
 - FULL PASS: 3 MINUTES PER 4K
 - ITERATION INHIBITED: 45 SECONDS PER 4K
- B. FOR PARITY MEMORY
 - FIRST PASS: 45 SECONDS +10 SECONDS PER 4K
 - FULL PASS: 4 MINUTES PER 4K
 - ITERATION INHIBITED: 1 MINUTE PER 4K

3.0 ERROR INFORMATION

317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372

3.1 ERROR REPORTING

THERE ARE A TOTAL OF 31(8) TYPES OF ERROR REPORTS GENERATED BY THE PROGRAM. SOME OF THE KEY COLUMN HEADING MNEMONICS ARE DESCRIBED BELOW FOR CLARITY:

PC = PROGRAM COUNTER OF ERROR DETECTION CODE.
(V/PC=P/PC)

V/PC = VIRTUAL PROGRAM COUNTER. THIS IS WHERE THE ERROR DETECTION CODE CAN BE FOUND IN THE PROGRAM LISTING.

P/PC = PHYSICAL PROGRAM COUNTER. THIS IS WHERE THE ERROR DETECTION CODE IS ACTUALLY LOCATED IN MEMORY.

TRP/PC = PHYSICAL PROGRAM COUNTER OF THE CODE WHICH CAUSED A TRAP.

MA = MEMORY ADDRESS

REG = PARITY REGISTER ADDRESS.

PS = PROCESSOR STATUS WORD.

IUT = INSTRUCTION UNDER TEST.

S/B = WHAT CONTENTS SHOULD BE.

WAS = WHAT CONTENTS WAS.

3.2 ERROR HALTS

WITH THE 'HALT ON ERROR' SWITCH (SW15) NOT SET THERE ARE SEVERAL PROGRAMMED 'HALTS' IN THE PROGRAM:

A. IN THE ERROR TRAP SERVICE ROUTINE FOR UNEXPECTED TRAPS TO VECTOR 4. THIS ONE WILL OCCUR IF A 2ND TRAP TO 4 OCCURS BEFORE THE ERROR REPORT FOR THE FIRST HAS HAD A CHANCE TO BE PRINTED OUT.

B. IN THE RELOCATION ROUTINE IF THE PROGRAM IS BEING RELOCATED BACK TO THE FIRST 8K OF MEMORY AND THE PROGRAM CODE WAS NOT ABLE TO BE TRANSFERRED PROPERLY.

C. IN THE CASE OF ERROR REPORTING AND THERE IS NO TERMINAL TO ALLOW THE INFORMATION TRANSFER.

D. IN THE POWER FAIL ROUTINE IF THE POWER UP SEQUENCE WAS STARTED BEFORE THE POWER DOWN SEQUENCE HAD A CHANCE TO COMPLETE ITSELF.

E. IN THE MEMORY MAPPING ROUTINE OR ANY OF THE ADDRESS

373

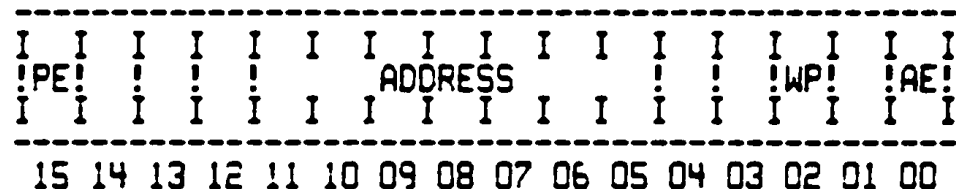
CONTROL ROUTINES. FAILURES TO FIND A MEANINGFUL MAP.

4.0 PERFORMANCE AND PROGRESS REPORTS

NOT APPLICABLE

5.0 DEVICE INFORMATION TABLES

THE FOLLOWING IS A PICTURE VIEW OF A PARITY CONTROL STATUS REGISTER, WHICH WILL SHOW BIT ASSIGNMENTS AND DEFINITIONS, TO PROVIDE A HANDY REFERENCE:



BIT ASSIGNMENTS ARE DEFINED AS FOLLOWS:

BIT15	PARITY ERROR	
BITS 11-5	ERROR ADDRESS	HIGH ORDER ADDRESS BITS OF ADDRESS OF PARITY ERROR (BITS 17-11 OF ADDRESS)
BIT02	WRITE WRONG PARITY	NORMAL PARITY (ODD) WHEN CLEAR; OTHER PARITY (EVEN) WHEN SET
BIT00	ACTION ENABLE	NO ACTION WHEN CLEAR TRAP TO VECTOR 114 WHEN SET

6.0 SUB-TEST SUMMARIES

6.1 SECTION 1: ADDRESS TESTS.

THESE TESTS VERIFY THE UNIQUENESS OF EVERY MEMORY ADDRESS.

TEST 1 WRITES AND READS THE VALUE OF EACH MEMORY WORD ADDRESS INTO THAT MEMORY LOCATION. AFTER ALL MEMORY HAS BEEN WRITTEN, ALL LOCATIONS ARE CHECKED AGAIN.

TEST 2 WRITES THE BYTE VALUE OF EACH ADDRESS INTO THAT BYTE LOCATION AND CHECKS IT.

TEST 3 WRITES THE COMPLEMENT OF EACH WORD ADDRESS INTO THAT LOCATION AND CHECKS IT.

4274
4275
4276
4277
4278
4279
4280
4281
4282
4283
4284
4285
4286
4287
4288
4289
4290
4291
4292
4293
4294
4295
4296
4297
4298
4299
4300
4301
4302
4303
4304
4305
4306
4307
4308
4309
4310
4311
4312
4313
4314
4315
4316
4317
4318
4319
4320
4321
4322
4323
4324
4325
4326
4327
4328
4329

802

MACDEC-11-D23MC-6-0: 0-124K MEMORY EXERCISER, 16K VER
2307CB.P11

MACY11 27(732) 10-SEP-76 12:01 PAGE 15

430
431

TEST 4 WRITES THE 4K BANK NUMBER INTO EACH BYTE OF THAT BANK
AND CHECKS IT.

002

MAINDEC-11-DZJMC-B-D: 0-124K MEMORY EXERCISER, 16K VER
DZJMCB.P11

MACY11 27(732) 10-SEP-76 12:01 PAGE 17

488
489

TEST 25 EXECUTES AN INSTRUCTION WHICH DOES A DATI AND A
DATOB ON THE HIGH BYTE.

TEST 26 EXECUTES AN INSTRUCTION WHICH DOES A DATIP AND A DATO.

TEST 27 EXECUTES AN INSTRUCTION WHICH DOES A DATIP AND A DATOB ON THE LOW BYTE.

TEST 30 EXECUTES AN INSTRUCTION WHICH DOES A DATIP AND A DATOB ON THE HIGH BYTE.

TEST 31 EXECUTES A SERIES OF INSTRUCTIONS DESIGNED TO STRESS OR "HEAT" THE MEMORY.

6.4 SPECIAL TOGGLE IN TESTS

6.4.1 TOGGLE-IN-PROGRAM #1

THE FOLLOWING IS A TOGGLE IN MEMORY ADDRESS TEST. THIS TEST IS USEFUL WHEN AN ADDRESS SELECTION FAILURE IS SUSPECTED INVOLVING THE FIRST 8K OF MEMORY. THIS PROGRAM WRITES THE VALUE OF EACH ADDRESS INTO ITSELF STARTING WITH THE LOWER LIMIT AND CONTINUING TO THE UPPER LIMIT. AFTER ALL ADDRESSES HAVE BEEN WRITTEN EACH ADDRESS IS CHECKED FOR THE CORRECT CONTENTS STARTING WITH THE UPPER LIMIT AND CONTINUING TO THE LOWER LIMIT.

LOCATION	CONTENTS	MNEMONIC	COMMENT
10	012700	MOV #50,R0	:GET FIRST ADDRESS
* 12	000050		:TO TEST :(EXAMPLE START ADDRESS)
14	010001	MOV R0,R1	:SAVE IN R1
16	020037	1\$: CMP R0,R#SWR	:CHECK UPPER LIMIT :(IN SWITCH REGISTER)
20	177570		
22	001403	BEG 2\$:BRANCH IF AT UPPER LIMIT
24	010010	MOV R0,(R0)	:LOAD VALUE INTO ADDRESS
26	005720	TST (R0)+	:STEP TO NEXT ADDRESS
30	000772	BR 1\$:LOOP UNTIL DONE
32	010004	2\$: MOV R0,R4	:SAVE UPPER LIMIT
34	020001	3\$: CMP R0,R1	:CHECK IF AT LOWER LIMIT
* 36	001767	BEG 1\$:BRANCH IF DONE
40	024000	CMP -(R0),R0	:CHECK DATA WRITTEN
42	001774	BEG 3\$:BRANCH IF OK
44	000000	HALT	:ERROR
46	000772	BR 3\$:LOOP BACK

AFTER TOGLING THE PROGRAM LA=10**SET UPPER LIMIT**, START

NOTES: THE UPPER LIMIT ADDRESS OBTAINED FROM THE SWITCH REGISTER MAY BE CHANGED DURING PROGRAM OPERATION. HOWEVER OCCASIONALLY THE PROGRAM MAY HALT BECAUSE OF 'SWITCH BOUNCE'. (THE BEST PROCEDURE WHEN CHANGING LIMITS IS TO STOP THE PROGRAM MAKE THE CHANGE AND

Vertical text on the left margin, likely a page number or document identifier, appearing as a series of characters.

546
547

CONTINUE.) THE LOWER LIMIT ADDRESS (12) MAY BE PATCHED
TO ANY DESIRED ADDRESS.

6.4.2 TOGGLE-IN-PROGRAM #2

THE FOLLOWING IS ALSO A TOGGLE IN PROGRAM TO BE USED WITH TOGGLE-IN-PROGRAM #1 FOR MORE COMPLETE ADDRESS TESTING. THIS PROGRAM WRITES THE COMPLEMENT VALUE OF EACH ADDRESS INTO ITSELF STARTING WITH THE UPPER LIMIT AND CONTINUING TO THE LOWER LIMIT. AFTER ALL ADDRESSES HAVE BEEN WRITTEN EACH ADDRESS IS CHECKED FOR THE CORRECT CONTENTS STARTING WITH THE LOWER LIMIT ADDRESS AND CONTINUING TO THE UPPER LIMIT. TOGGLE IN THE FOLLOWING PATCHES TO THE PROGRAM ABOVE.

THESE ARE THE PATCHES TO TOGGLE-IN-PROGRAM #1:

LOCATION	CONTENTS	MNEMONIC	COMMENT
12	100		:CHANGE LOWER LIMIT
36	001404	BEQ 4\$:BRANCH TO PROGRAM #2

THESE ARE THE ADDITIONS TO TOGGLE-IN-PROGRAM #1:

LOCATION	CONTENTS	MNEMONIC	COMMENT
50	010402	4\$: MOV R4, R2	:GET UPPER LIMIT
52	005142	5\$: COM -(R2)	:COMPLEMENT ADDRESS
54	020201	CMP R2, R1	:CHECK IF AT LOWER LIMIT
56	001375	BNE 5\$:LOOP UNTIL DONE
60	020204	6\$: CMP R2, R4	:CHECK IF AT UPPER LIMIT
62	001755	BEQ 1\$:GO TO PROGRAM 1 IF DONE
64	010203	MOV R2, R3	:GET VALUE OF ADDRESS
66	005103	COM R3	:COMPLEMENT VALUE
70	020322	CMP R3, (R2)+	:CHECK ADDRESS
72	001772	BEQ 6\$:BRANCH IF OK
74	000000	HALT	:ERROR
76	000770	BR 6\$:GO CHECK NEXT ADDRESS

7.0 PROGRAM FUNCTIONAL FLOW CHARTS
ATTACHED

8.0 PROGRAM LISTING
ATTACHED

548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592

591

%

```
.TITLE MAINDEC-11-DZQMC-B-D: 0-124K MEMORY EXERCISER, 16K VER
.*COPYRIGHT (C) 1975
.*DIGITAL EQUIPMENT CORP.
.*MAYNARD, MASS. 01754
.*
.*PROGRAM BY BRUCE BURGESS, KEN CHAPMAN
.*
.*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
.*PACKAGE (MAINDEC-11-DZQAC-B1), AUG 29, 1975.
.*
```

604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646

```
.SBTTL OPERATIONAL SWITCH SETTINGS
.*
.* SWITCH USE
.* -----
.* 15 HALT ON ERROR
.* 14 LOOP ON TEST
.* 13 INHIBIT ERROR TYPEOUTS
.* 12 INHIBIT KT11 (AT START TIME ONLY)
.* 11 INHIBIT ITERATIONS
.* 10 BELL ON ERROR
.* 9 LOOP ON ERROR
.* 8 LOOP ON TEST IN SWR<4:0>
.* 7 INHIBIT PROGRAM RELOCATION
.* 6 INHIBIT PARITY ERROR DETECTION
.* 5 INHIBIT EXERCISING VECTOR AREA.

.SBTTL BASIC DEFINITIONS

.*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
001100 STACK= 1100
.EQUIV EMT,ERROR ;;BASIC DEFINITION OF ERROR CALL
.EQUIV IOT,SCOPE ;;BASIC DEFINITION OF SCOPE CALL
177776 PS= 177776 ;;PROCESSOR STATUS WORD
.EQUIV PS,PSW
177774 STKLM= 177774 ;;STACK LIMIT REGISTER
177772 PIRQ= 177772 ;;PROGRAM INTERRUPT REQUEST REGISTER
177570 DSWR= 177570 ;;HARDWARE SWITCH REGISTER
177570 DDISP= 177570 ;;HARDWARE DISPLAY REGISTER

.*GENERAL PURPOSE REGISTER DEFINITIONS
R0= %0 ;;GENERAL REGISTER
R1= %1 ;;GENERAL REGISTER
R2= %2 ;;GENERAL REGISTER
R3= %3 ;;GENERAL REGISTER
R4= %4 ;;GENERAL REGISTER
R5= %5 ;;GENERAL REGISTER
R6= %6 ;;GENERAL REGISTER
R7= %7 ;;GENERAL REGISTER
.EQUIV R6,SP ;;STACK POINTER
.EQUIV R7,PC ;;PROGRAM COUNTER

.*PRIORITY LEVEL DEFINITIONS
```

647 000000
 648 000040
 649 000100
 650 000140
 651 000200
 652 000240
 653 000300
 654 000340

PR0= 0
 PR1= 40
 PR2= 100
 PR3= 140
 PR4= 200
 PR5= 240
 PR6= 300
 PR7= 340

:: PRIORITY LEVEL 0
 :: PRIORITY LEVEL 1
 :: PRIORITY LEVEL 2
 :: PRIORITY LEVEL 3
 :: PRIORITY LEVEL 4
 :: PRIORITY LEVEL 5
 :: PRIORITY LEVEL 6
 :: PRIORITY LEVEL 7

655
 656 100000
 657 040000
 658 020000
 659 010000
 660 004000
 661 002000
 662 001000
 663 000400
 664 000200
 665 000100
 666 000040
 667 000020
 668 000010
 669 000004
 670 000002
 671 000001

.*"SWITCH REGISTER" SWITCH DEFINITIONS

SW15= 100000
 SW14= 40000
 SW13= 20000
 SW12= 10000
 SW11= 4000
 SW10= 2000
 SW09= 1000
 SW08= 400
 SW07= 200
 SW06= 100
 SW05= 40
 SW04= 20
 SW03= 10
 SW02= 4
 SW01= 2
 SW00= 1

.EQUIV SW09, SW9
 .EQUIV SW08, SW8
 .EQUIV SW07, SW7
 .EQUIV SW06, SW6
 .EQUIV SW05, SW5
 .EQUIV SW04, SW4
 .EQUIV SW03, SW3
 .EQUIV SW02, SW2
 .EQUIV SW01, SW1
 .EQUIV SW00, SW0

672
 673
 674
 675
 676
 677
 678
 679
 680
 681
 682
 683
 684
 685 100000
 686 040000
 687 020000
 688 010000
 689 004000
 690 002000
 691 001000
 692 000400
 693 000200
 694 000100
 695 000040
 696 000020
 697 000010
 698 000004
 699 000002
 700 000001

.*DATA BIT DEFINITIONS (BIT00 TO BIT15)

BIT15= 100000
 BIT14= 40000
 BIT13= 20000
 BIT12= 10000
 BIT11= 4000
 BIT10= 2000
 BIT09= 1000
 BIT08= 400
 BIT07= 200
 BIT06= 100
 BIT05= 40
 BIT04= 20
 BIT03= 10
 BIT02= 4
 BIT01= 2
 BIT00= 1

.EQUIV BIT09, BIT9
 .EQUIV BIT08, BIT8

701
 702

```

703 .EQUIV BIT07,BIT7
704 .EQUIV BIT06,BIT6
705 .EQUIV BIT05,BIT5
706 .EQUIV BIT04,BIT4
707 .EQUIV BIT03,BIT3
708 .EQUIV BIT02,BIT2
709 .EQUIV BIT01,BIT1
710 .EQUIV BIT00,BIT0

711
712 ;*BASIC "CPU" TRAP VECTOR ADDRESSES
713 000004 ERRVEC= 4 ;: TIME OUT AND OTHER ERRORS
714 000010 RESVEC= 10 ;: RESERVED AND ILLEGAL INSTRUCTIONS
715 000014 TBITVEC=14 ;: "T" BIT
716 000014 TRTVEC= 14 ;: TRACE TRAP
717 000014 BPTVEC= 14 ;: BREAKPOINT TRAP (BPT)
718 000020 IOTVEC= 20 ;: INPUT/OUTPUT TRAP (IOT) **SCOPE**
719 000024 PWRVEC= 24 ;: POWER FAIL
720 000030 EMTVEC= 30 ;: EMLLATOR TRAP (EMT) **ERROR**
721 000034 TRAPVEC=34 ;: "TRAP" TRAP
722 000060 TKVEC= 60 ;: TTY KEYBOARD VECTOR
723 000064 TPVEC= 64 ;: TTY PRINTER VECTOR
724 000240 PIRQVEC=240 ;: PROGRAM INTERRUPT REQUEST VECTOR
725
726
727
728 .SBTTL MEMORY MANAGEMENT DEFINITIONS
729
730 ;*KT11 VECTOR ADDRESS
731
732 000250 MMVEC= 250
733
734 ;*KT11 STATUS REGISTER ADDRESSES
735
736 177572 SR0= 177572
737 177574 SR1= 177574
738 177576 SR2= 177576
739 172516 SR3= 172516
740
741 ;*KERNAL "I" PAGE DESCRIPTOR REGISTERS
742
743 172300 KIPDR0= 172300
744 172302 KIPDR1= 172302
745 172304 KIPDR2= 172304
746 172306 KIPDR3= 172306
747 172310 KIPDR4= 172310
748 172312 KIPDR5= 172312
749 172314 KIPDR6= 172314
750 172316 KIPDR7= 172316
751
752 ;*KERNAL "I" PAGE ADDRESS REGISTERS
753
754 172340 KIPAR0= 172340
755 172342 KIPAR1= 172342
756 172344 KIPAR2= 172344
757 172346 KIPAR3= 172346
758 172350 KIPAR4= 172350
    
```

```

759          172352      KIPAR5= 172352
760          172354      KIPAR6= 172354
761          172356      KIPAR7= 172356
762          000000      UP = 0          ;CODE FOR UPWARDS MAP IN MEM MGMT PDR'S
763          000006      RW = 6          ;CODE FOR READ/WRITE IN MEM MGMT PDR'S
764
765
766          000004      ;* PARITY MEMORY DEFINITIONS.
767          000000      WWP=4          ;WRITE WRONG PARITY
768          000001      AE=1          ;PARITY ACTION ENABLE
769          000114      PARVEC=114      ;PARITY TRAP VECTOR
770
771          ;* MISCELLANEOUS ASSIGNMENTS
772          017777      MASK4K= 17777      ;MASK FOR 4K ADDRESS BANK BOUNDRY.
773
774          .SBTTL TRAP CATCHER
775
776          000000      .=0
777
778          ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2.HALT"
779          ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
780          ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
781          000174      .=174
782          000174      000000      DISPREG: .WORD 0          ;;SOFTWARE DISPLAY REGISTER
783          000176      000000      SWREG:   .WORD 0          ;;SOFTWARE SWITCH REGISTER
784
785          .SBTTL STARTING ADDRESS(ES)
786          000200      000137      002624      JMP      @#START      ;;JUMP TO STARTING ADDRESS OF PROGRAM
787          000204      000167      002422      JMP      SELECT      ;;STARTING ADDRESS TO ALLOW THE OPERATOR TO
788                                     ;;SELECT VARIOUS PARAMETERS.
789          000210      000167      000064      JMP      RESTAR      ;;RESTART ADDRESS, USING PREVIOUS PARAMETERS.
790          000214      000167      000064      JMP      RESTOR      ;;RESTORE LOADERS TO END OF MEMORY AND HALT.
791          000220      000167      003272      JMP      TIMEOUT     ;;TYPE OUT MEMORY MAP, BYTE BY BYTE.
792
793          000004      .=ERRVEC
794          000004      024636      .WORD  ERRTRP
795          000006      000000      .WORD  0
796
797          ;*****
798
799          .SBTTL ACT11 HOOKS
800          ;HOOKS REQUIRED BY ACT11
801          000010      $SVPC=.          ;SAVE PC
802          000046      .=46
803          000046      014546      $ENDAD          ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
804          000052      .=52
805          000052      040000      .WORD  BIT14      ;;2)SET LOC.52 TO BIT14
806          000010      .=$SVPC          ;; RESTORE PC

```


807 J00300

.=300

808
809
810
811
812
813
814 000300 005005
815 000302 000401
816 000304 010705
817 000306 012706 001100
818 000312 005767 001204
819 000316 001002
820 000320 000167 002314
821 000324 005767 000256
822 000330 001470
823 000332 032737 000001 177572
824 000340 001034
825 000342 012700 172300
826 000346 012701 000010
827 000352 012720 077406
828 000356 005301
829 000360 001374
830 000362 012700 172340
831 000366 005020
832 000370 012720 000200
833 000374 012720 000400
834 000400 012720 000600
835 000404 012720 001000
836 000410 012720 001200
837 000414 012720 001400
838 000420 012720 007600
839 000424 012737 000001 177572
840 000432 005000
841 000434 016701 000142
842 000440 016702 000140
843 000444 006202
844 000446 006001
845 000450 103404
846 000452 062700 000200
847 000456 100372
848 000460 000000
849 000462 010037 172340
850 000466 000137 000472
851 000472 062700 000200
852 000476 006202
853 000500 006001
854 000502 103373
855 000504 010037 172342
856 000510 000410
857 000512 016700 000062
858 000516 062700 001100
859 000522 010006
860 000524 062700 177432
861 000530 000110
862 000532 022767 000003 000042

```

*****
* THE FOLLOWING ROUTINES ARE LOCATED IN THE VECTOR AREA (0-1000) SO THAT
* THEY CAN BE PROTECTED BY SELECTING SW05 (SEE DOCUMENT FOR USE OF SW05).
* THE CODE CAN ALSO BE RUN FROM ANY BANK OF MEMORY, ASSUMING MEMORY
* MANAGEMENT IS DISABLED BY "CONSOLE START".
*****
RESTAR: CLR R5 ;CLEAR FLAG TO INDICATE RESTART.
          BR REST1 ;GO RESTORE PROGRAM BEFORE RESTARTING.
RESTOR: MOV PC, R5 ;PUT DATA INTO FLAG FOR RESTORE.
REST1: MOV #STACK, SP ;SET UP THE STACK POINTER.
          TST MEMMAP ;CHECK IF THE MEMORY HAS BEEN MAPPED.
          BNE REST2 ;BR IF MEMORY MAPPED.
          JMP STARTA ;GO START
REST2: TST MMAPA ;CHECK IF MEM MGMT AVAILABLE.
          BEQ 10$ ;BR IF NO MEM MGMT.
          BIT #BIT0, @#SRO ;CHECK IF MEM MGMT ACTIVE.
          BNE 2$ ;BR IF MEM MGMT ALREADY SET UP.
          MOV #KIPDR0, R0 ;POINT TO FIRST MEM MGMT DDATA REG.
          MOV #8, R1 ;SET UP COUNTER.
1$: MOV #077406, (R0)+ ;MAP FIRST 28K 1-FOR-1.
          DEC R1 ;COUNT REGESTERS.
          BNE 1$ ;BR IF MORE REG.
          MOV #KIPAR0, R0 ;POINT TO FIRST MEM MGMT ADDRESS REG.
          CLR (R0)+ ;PAR0 MAPPED INTO BANK0.
          MOV #200, (R0)+ ;PAR1 MAPPED INTO BANK1.
          MOV #400, (R0)+ ;PAR2 MAPPED INTO BANK2.
          MOV #600, (R0)+ ;PAR3 MAPPED INTO BANK3.
          MOV #1000, (R0)+ ;PAR4 MAPPED INTO BANK4.
          MOV #1200, (R0)+ ;PAR5 MAPPED INTO BANK5.
          MOV #1400, (R0)+ ;PAR6 MAPPED INTO BANK6.
          MOV #7600, (R0)+ ;PAR7 MAPPED INTO BANK37.
          MOV #BIT0, @#SRO ;ENABLE MEM MGMT.
2$: CLR R0 ;INIT TEMP PAR REG.
          MOV PRGMAP, R1 ;GET THE PROGRAM MAP...LO 64K.
          MOV PRGMAP+2, R2 ;...HI 64K.
3$: ASR R2 ;SHIFT THE MAP POINTER...HI
          ROR R1 ;...LO.
          BCS 4$ ;BR WHEN FIRST BANK FOUND.
          ADD #200, R0 ;UPDATE TMP PAR TO NEXT BANK.
          BPL 3$ ;BR IF MORE.
          HALT ;FATAL ERROR!!! MAP EMPTY?
4$: MOV R0, @#KIPAR0 ;PUT TEMP PAR INTO FIRST PAR.
          JMP @#5$ ;JUMP INTO PROGRAM IF NOT THERE ALREADY.
5$: ADD #200, R0 ;KEEP UPDATING TEMP PAR REG.
          ASR R2 ;SHIFT POINTER...HI
          ROR R1 ;...LO
          BCC 5$ ;BR IF TOP BANK NOT YET FOUND.
          MOV R0, @#KIPAR1 ;SET UP SECOND PROGRAM ANK POINTER.
          BR 20$ ;BR TO RELOCATE SECTION.
10$: MOV RELOCF, R0 ;GET RELOCATION FACTOR.
          ADD #STACK, R0 ;SET UP STACK POINTER.
          MOV R0, SP ;SET STACK TO RELOCATE PROGRAM.
          ADD #20$-STACK, R0 ;ADJUST R0 TO RELOCATED "20$" ADDRESS.
          JMP (R0) ;GO TO "20$" (RELOCATED).
20$: CMP #3, PRGMAP ;CHECK IF PROGRAM IS IN BANKS 0 AND 1.

```

M02

MAINDEC-11-DZQMC-B-D: 0-124K MEMORY EXERCISER, 16K VER
DZQMCB.P11 ACT11 HOOKS

MACY11 27(732) 10-SEP-76 12:01 PAGE 26

863	000540	001402	
864	000542	004767	016616
865	000546	005705	
866	000550	001006	
867	000552	005067	000410
868	000556	105067	000320
869	000562	000167	004576
870	000566	004767	017000
871	000572	000000	
872	000574	000167	002040
873			
874			
875			
876	000600	000000	
877	000602	000000	000000
878	000606	000000	

```

21$: BEQ 21$ ;BR IF IN BANKS 0 AND 1.
      JSR PC, RELO ;RELOCATE THE PROGRAM BACK TO BANKS 0 AND 1.
      TST R5 ;CHECK RESTART/RESTORE FLAG.
      BNE 22$ ;BP IF RESTORE.
      CLR $TIMES ;CLEAN UP BEFORE STARTING.
      CLRB $STNM
22$: JMP START1 ;RESTART WITH PREVIOUSLY SELECTED PARAMETERS.
      JSR PC, RESLDR ;RESTORE THE LOADERS TO THE "TOP" OF MEMORY.
      HALT ;HALT AFTER RESTORING THE LOADERS.
      JMP STARTA ;CONTINUE WILL RESTART THE PROGRAM.
; * THE FOLLOWING LOCATIONS ARE USED BY THE ABOVE ROUTINE AND MUST BE LOCATED
; * BELOW 1000 TO INSURE CORRECT OPERATION UNDER THE WIDEST VARIETY OF
; * CIRCUMSTANCES.
RELOCF: .WORD 0 ;CONTAINS RELOCATION FACTOR (NO MEM MGMT)
PRGMAP: .WORD 0,0 ;PROGRAM MAP - WHERE THE PROGRAM IS LOCATED
MMAVA: .WORD 0 ;MEMORY MANAGEMENT AVAILABLE FLAG.

```

```

879                                     :*****
880                                     :
881 .SBTTL POWER DOWN AND UP ROUTINES
882
883 :POWER DOWN ROUTINE
884 000610 012737 000740 000024 $PWRDN: MOV $SILLUP, @#PWRVEC ;; SET FOR FAST UP
885 000616 012737 000340 000026 MOV #340, @#PWRVEC+2 ;; PRIO:7
886 000624 010046 MOV R0, -(SP) ;; PUSH R0 ON STACK
887 000626 010146 MOV R1, -(SP) ;; PUSH R1 ON STACK
888 000630 010246 MOV R2, -(SP) ;; PUSH R2 ON STACK
889 000632 010346 MOV R3, -(SP) ;; PUSH R3 ON STACK
890 000634 010446 MOV R4, -(SP) ;; PUSH R4 ON STACK
891 000636 010546 MOV R5, -(SP) ;; PUSH R5 ON STACK
892 000640 010667 000100 MOV SP, $SAVR6 ;; SAVE SP
893 000644 012737 000656 000024 MOV $PWRUP, @#PWRVEC ;; SET UP VECTOR
894 000652 000000 HALT
895 000654 000776 BR .-2 ;; HANG UP
896
897 :POWER UP ROUTINE
898 000656 016706 000062 $PWRUP: MOV $SAVR6, SP ;; GET SP
899 000662 005067 000056 CLR $SAVR6 ;; WAIT LOOP FOR THE TTY
900 000666 005267 000052 1$: INC $SAVR6 ;; WAIT FOR THE INC
901 000672 001375 BNE 1$ ;; OF WORD
902 000674 012605 MOV (SP)+, R5 ;; POP STACK INTO R5
903 000676 012604 MOV (SP)+, R4 ;; POP STACK INTO R4
904 000700 012603 MOV (SP)+, R3 ;; POP STACK INTO R3
905 000702 012602 MOV (SP)+, R2 ;; POP STACK INTO R2
906 000704 012601 MOV (SP)+, R1 ;; POP STACK INTO R1
907 000706 012600 MOV (SP)+, R0 ;; POP STACK INTO R0
908 000710 012737 000610 000024 MOV $PWRDN, @#PWRVEC ;; SET UP THE POWER DOWN VECTOR
909 000716 012737 000340 000026 MOV #340, @#PWRVEC+2 ;; PRIO:7
910 000724 004567 022310 JSR R5, $PRINT ;; GO PRINT OUT THE FOLLOWING MESSAGE.
911 000730 025363 $PWARMG: .WORD PWARMG ;; POWER FAIL MESSAGE POINTER
912 000732 012716 MOV (PC)+, (SP) ;; RESTART AT RESTART
913 000734 000300 $PWRAD: .WORD RESTART ;; RESTART ADDRESS
914 000736 000002 RTI
915 000740 000000 $SILLUP: HALT ;; THE POWER UP SEQUENCE WAS STARTED
916 000742 000776 BR .-2 ;; BEFORE THE POWER DOWN WAS COMPLETE
917 000744 000000 $SAVR6: 0 ;; PUT THE SP HERE

```

.SBTTL COMMON TAGS

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

```

001100
001102 000000
001103 000
001104 000
001105 000000
001106 000000
001108 000000
001109 000000
001110 000000
001111 000000
001112 000000
001113 000
001114 001
001115 000000
001116 000000
001117 000000
001118 000000
001119 000000
001120 000000
001121 000000
001122 000000
001123 000000
001124 000000
001125 000000
001126 000000
001127 000000
001128 000000
001129 000000
001130 000000
001131 000000
001132 000000
001133 000000
001134 000000
001135 177570
001136 177570
001137 177560
001138 177560
001139 177562
001140 177562
001141 177564
001142 177564
001143 177566
001144 177566
001145 000
001146 000
001147 012
001148 000
001149 000
001150 000000
001151 000000
001152 000000
001153 000000
001154 000000
001155 000000
001156 000000
001157 000000
001158 000000
001159 000000
001160 000000
001161 000000
001162 000000
001163 000000
001164 000000
001165 000000
001166 000000
001167 000000
001168 177607
001169 000377
001170 077
001171 015
001172 000012
001173
001174
001175
001176
001177
001178
001179
001180
001181
001182
001183
001184
001185
001186
001187
001188
001189
001190
001191
001192
001193
001194
001195
001196
001197
001198
001199
001200

```

```

SCMTAG:    =1100
           .WORD    0
STSTNM:    .BYTE   0000
SERFLG:    .BYTE   0000
SICNT:     .WORD   000000
SLPADR:    .WORD   000000
SLPERR:    .WORD   000000
SERTTL:    .WORD   000000
SITEMB:    .BYTE   000
SERMAX:    .BYTE   001
SERAPC:    .WORD   000000
SGOADR:    .WORD   000000
SBDADR:    .WORD   000000
SGDDAT:    .WORD   000000
SBDAT:     .WORD   000000
           .WORD   000000
           .WORD   000000
           .WORD   000000
SWR:       .WORD   DSWR
DISPLAY:   .WORD   DDISP
STKS:      177560
STKB:      177562
STPS:      177564
STPB:      177566
SNULL:     .BYTE   0
SFILLS:    .BYTE   0
SFILLC:    .BYTE   0
STPFLG:    .BYTE   0
STMP0:     .WORD   0
STMP1:     .WORD   0
STMP2:     .WORD   0
STMP3:     .WORD   0
STIMES:    0
SESCAPE:   0
SBELL:     .ASCIZ  ?07<<377><377>
SQUES:     .ASCTI  ?
SCRLF:     .ASCTI  \15\
SLF:       .ASCIZ  \12\

```

```

:: START OF COMMON TAGS
:: CONTAINS THE TEST NUMBER
:: CONTAINS ERROR FLAG
:: CONTAINS SUBTEST ITERATION COUNT
:: CONTAINS SCOPE LOOP ADDRESS
:: CONTAINS SCOPE RETURN FOR ERRORS
:: CONTAINS TOTAL ERRORS DETECTED
:: CONTAINS ITEM CONTROL BYTE
:: CONTAINS MAX. ERRORS PER TEST
:: CONTAINS PC OF LAST ERROR INSTRUCTION
:: CONTAINS ADDRESS OF 'GOOD' DATA
:: CONTAINS ADDRESS OF 'BAD' DATA
:: CONTAINS 'GOOD' DATA
:: CONTAINS 'BAD' DATA
:: RESERVED--NOT TO BE USED

:: ADDRESS OF SWITCH REGISTER
:: ADDRESS OF DISPLAY REGISTER
:: TTY KBD STATUS
:: TTY KBD BUFFER
:: TTY PRINTER STATUS REG. ADDRESS
:: TTY PRINTER BUFFER REG. ADDRESS
:: CONTAINS NULL CHARACTER FOR FILLS
:: CONTAINS # OF FILLER CHARACTERS REQUIRED
:: INSERT FILL CHARS. AFTER A "LINE FEED"
:: "TERMINAL AVAILABLE" FLAG (BIT 07)=0=NO, 1=YES
:: USER DEFINED
:: USER DEFINED
:: USER DEFINED
:: USER DEFINED
:: MAX. NUMBER OF ITERATIONS
:: ESCAPE ON ERROR ADDRESS
:: CODE FOR BELL
:: QUESTION MARK
:: CARRIAGE RETURN
:: LINE FEED

```

964
 965
 966
 967
 968
 969
 970
 971
 972
 973
 974
 975
 976
 977
 978
 979
 980
 981
 982
 983
 984
 985
 986
 987
 988
 989
 990
 991
 992
 993
 994
 995
 996
 997
 998
 999
 1000
 1001
 1002
 1003
 1004
 1005
 1006
 1007
 1008
 1009
 1010
 1011
 1012
 1013
 1014
 1015
 1016
 1017
 1018
 1019

.SBTTL APT MAILBOX-ETABLE

```

.EVEN
$MAIL:          .WORD      AMSGTY  :: APT MAILBOX
$MSGTY:         .WORD      AFATAL   :: MESSAGE TYPE CODE
$FATAL:         .WORD      ATESTN   :: FATAL ERROR NUMBER
$TESTN:         .WORD      APASS     :: TEST NUMBER
$PASS:          .WORD      ADEVCT   :: PASS COUNT
$DEVCT:         .WORD      AUNIT    :: DEVICE COUNT
$UNIT:          .WORD      AMSGAD   :: I/O UNIT NUMBER
$MSGAD:         .WORD      AMSGLG   :: MESSAGE ADDRESS
$MSGLG:         .WORD      ASETABLE :: MESSAGE LENGTH
$ETABLE:        .WORD      AENV     :: APT ENVIRONMENT TABLE
$ENV:           .BYTE      AENVM    :: ENVIRONMENT BYTE
$ENVM:          .BYTE      ASWREG   :: ENVIRONMENT MODE BITS
$SWREG:         .WORD      AUSWR    :: APT SWITCH REGISTER
$USWR:          .WORD      ACPJOP   :: USER SWITCHES
$CPJOP:         .WORD      BIT15-11:: CPU TYPE, OPTIONS
                *          BIT 15-11=CPU TYPE
                *          11/04=01, 11/05=02, 11/20=03, 11/40=04, 11/45=05
                *          11/70=06, PDQ=07, Q=10
                *          BIT 10=REAL TIME CLOCK
                *          BIT 9=FLOATING POINT PROCESSOR
                *          BIT 8=MEMORY MANAGEMENT
$MAMS1:         .BYTE      AMAMS1   :: HIGH ADDRESS, M.S. BYTE
$MTYP1:         .BYTE      AMTYP1   :: MEM. TYPE, BLK#1
                *          MEM. TYPE BYTE -- (HIGH BYTE)
                *          900 NSEC CORE=001
                *          300 NSEC BIPOLAR=002
                *          500 NSEC MOS=003
$MADR1:         .WORD      AMADR1   :: HIGH ADDRESS, BLK#1
                *          MEM. LAST ADDR.=3 BYTES, THIS WORD AND LOW OF "TYPE" ABOVE
$MAMS2:         .BYTE      AMAMS2   :: HIGH ADDRESS, M.S. BYTE
$MTYP2:         .BYTE      AMTYP2   :: MEM. TYPE, BLK#2
$MADR2:         .WORD      AMADR2   :: MEM. LAST ADDRESS, BLK#2
$MAMS3:         .BYTE      AMAMS3   :: HIGH ADDRESS, M.S. BYTE
$MTYP3:         .BYTE      AMTYP3   :: MEM. TYPE, BLK#3
$MADR3:         .WORD      AMADR3   :: MEM. LAST ADDRESS, BLK#3
$MAMS4:         .BYTE      AMAMS4   :: HIGH ADDRESS, M.S. BYTE
$MTYP4:         .BYTE      AMTYP4   :: MEM. TYPE, BLK#4
$MADR4:         .WORD      AMADR4   :: MEM. LAST ADDRESS, BLK#4
$VECT1:         .BYTE      AVECT1   :: INTERRUPT VECTOR#1
$VECT2:         .BYTE      AVECT2   :: INTERRUPT VECTOR#2
$SPRIOR:        .BYTE      APRIOR   :: BUS PRIORITY #1, #2
                .BYTE      0        :: SPARE, NOT USED
                .EVEN
$BASE:          .WORD      ABASE    :: BASE ADDRESS OF EQUIPMENT UNDER TEST
$LEVM:         .WORD      ADEV     :: DEVICE MAP
$CDW1:         .WORD      ACDW1    :: CONTROLLER DESCRIPTION WORD#1
$CDW2:         .WORD      ACDW2    :: CONTROLLER DESCRIPTION WORD#2
$DDWC:         .WORD      ADDWC    :: DEVICE DESCRIPTOR WORD#0
  
```

25

E03

1076	001426	177777	000000	.WORD	-1.0
1077	001432	177777	000000	.WORD	-1.0
1078	001436	177777	000000	.WORD	-1.0
1079	001442	177777	000000	.WORD	-1.0
1080	001446	177777	000000	.WORD	-1.0
1081	001452	177777	000000	.WORD	-1.0
1082	001456	177777	000000	.WORD	-1.0
1083	001462	177777	000000	.WORD	-1.0
1084	001466	177777	000000	.WORD	-1.0
1085	001472	177777	000000	.WORD	-1.0
1086	001476	177777	000000	.WORD	-1.0
1087	001502	177777	000000	.WORD	-1.0
1089	001506	177777	000000	.WORD	-1.0
1099	001510	001342		\$ASTEND:	-1
				\$APTR:	\$ASTAT

```

:*****
:*THE FOLLOWING TAGS ARE USER DEFINED
:*****
$VERPC: .WORD 0 ;VIRTUAL PC LOCATION FOR ERROR TYPEOUT ROUTINE ($ERTYP).
RESRVD: .WORD 070032 ;CORE PARITY REG BITS RESERVED FOR FUTURE USE.
;NOTE: FOR MS11 MEMORY WITH PARITY, CHANGE TO 077772.
LMAD: .WORD 0 ;LAST CONTIGUOUS MEMORY ADDRESS (+2)
LDDISP: .WORD 0 ;CONTAINS DISPLAY REGISTER IMAGE
MEMMAP: .WORD 0 ;MEMORY MAP - EACH BIT CORRESPONDS TO 4K
;FIRST WORD CONTAINS LOW (0-64K) MAP
;SECOND WORD CONTAINS HIGH (64-128K) MAP
TSTMAP: .WORD 0 ;TEST MAP - WHICH BANKS ARE SELECTED FOR TEST.
;FIRST WORD CONTAINS LOW (0-64K) MAP
;SECOND WORD CONTAINS HIGH (64-128K) MAP
SAVTST: .WORD 0 ;SAVED TEST MAP - USED DURING FIRST PASS TO ONLY
;TEST EACH BANK ONCE.
;FIRST WORD CONTAINS LOW (0-64K) MAP
;SECOND WORD CONTAINS HIGH (64-128K) MAP
PMEMAP: .WORD 0 ;PARITY MAP - WHICH BANKS HAVE MEMORY PARITY
;FIRST WORD CONTAINS LOW (0-64K) MAP
;SECOND WORD CONTAINS HIGH (64-128K) MAP
BITPT: .WORD 0 ;POINTER TO CURRENT 4K BANK OF MEMORY
;FIRST WORD CONTAINS LOW (0-64K) MAP
;SECOND WORD CONTAINS HIGH (64-128K) MAP
TMPPT: .WORD 0 ;TEMPORARY POINTER FOR 2ND 4K BANK OF MEMORY
;FIRST WORD CONTAINS LOW (0-64K) MAP
;SECOND WORD CONTAINS HIGH (64-128K) MAP
MMORE: .WORD 0 ;LOOP ADDRESS FOR MULTIPLE BLOCK TESTING.
;SET UP BY "INITMM" AND "INITDN" ROUTINES.
;USED BY "MMUP" AND "MMDOWN" ROUTINES.
SELFLG: .BYTE 0 ;OPERATOR SELECTED PARAMETERS FLAG. (SA=204
FLAGBK: .BYTE 0 ;8K BLOCK INDICATOR. USED IN "INITMM" AND "MMUP".
OEFLG: .BYTE 0 ;ODD/EVEN FLAG USED IN PARITY MEMORY BYTE TEST.
;EVEN
FSTADR: .WORD 0 ;FIRST VIRTUAL ADDRESS TO BE TESTED.
;FIRST ADDRESS IS USER SELECTABLE.
TMPFAD: .WORD 0 ;ADJUSTED FIRST ADDRESS.
FADMSK: .WORD 0 ;BIT MASK TO ALLOW DOWNWARD ADDRESSING TESTS
;TO BREAK TO "MMDOWN" TO FIND FIRST ADDRESS.
FADMAP: .WORD 0,C ;MAP OF BANK IN WHICH FIRST ADDRESS IS LOCATED.
LSTADR: .WORD 0 ;LAST VIRTUAL ADDRESS (+2) TO BE TESTED.

```

1090					
1091					
1092					
1093					
1094	001512	000000			
1095	001514	070032			
1096					
1097	001516	000000			
1098	001520	000000			
1099	001522				
1100	001522	000000			
1101	001524	000000			
1102	001526				
1103	001526	000000			
1104	001530	000000			
1105	001532				
1106					
1107	001532	000000			
1108	001534	000000			
1109	001536				
1110	001536	000000			
1111	001540	000000			
1112	001542				
1113	001542	000000			
1114	001544	000000			
1115	001546				
1116	001546	000000			
1117	001550	000000			
1118	001552	000000			
1119					
1120					
1121	001554	000			
1122	001555	000			
1123	001556	000			
1124		001560			
1125	001560	000000			
1126					
1127	001562	000000			
1128	001564	000000			
1129					
1130	001566	000000	000000		
1131	001572	000000			

```

1132                                     ;LAST ADDRESS IS USER SELECTABLE.
1133 001574 000000 TEMPLAD: .WORD 0 ;ADJUSTED LAST ADDRESS.
1134 001576 000000 LACMSK: .WORD 0 ;BIT MASK TO ALLOW UPWARD ADDRESSING TESTS
1135                                     ; TO BREAK TO "MMUP" TO FIND LAST ADDRESS.
1136 001600 000000 000000 LADMAP: .WORD 0,0 ;MAP OF BANK IN WHICH LAST ADDRESS IS LOCATED.
1137 001604 000000 BLKMSK: .WORD 0 ;BLOCK MASK, DETERMINES THE BLOCK SIZE.
1138 001606 000000 .CONST: .WORD 0 ;USER SELECTABLE CONSTANT DATA.
1139
1140 ;*****
1141 ;* RELATIVE ADDRESSING TABLE.
1142 ;* THE FOLLOWING LOCATIONS ARE MODIFIED AT RELOCATION TIME TO ALLOW
1143 ;* RELATIVE ADDRESSING TO GET THE RELOCATED VALUE OF THE ARGUMENT TAGS.
1144 ;*****
1145 RADTAB:
1146 001610 001100 .STACK: STACK ;STACK POINTER INITIAL ADDRESS.
1147 001612 001514 .RESRV: RESRVD ;PARITY REGISTER RESERVED BIT MASK ADDRESS.
1148 001614 002112 .MPRO: MPRO ;MEMORY PARITY REGISTER TABLE ADDRESS.
1149 001616 002252 .MPRX: MPRX ;MEMORY PARITY REGISTER EXIST TABLE ADDRESS.
1150 001620 014124 .BRGOB: BRGOB ;"BRANCH GOBBLE" ROUTINE ADDRESS.
1151 001622 014176 .BGERR: BGERR ;"BRANCH GOBBLE" ERROR ROUTINE ADDRESS.
1152 001624 014310 .BGEXI: BGEXI ;"BRANCH GOBBLE" EXIT ROUTINE ADDRESS.
1153 001626 012456 .PBTRP: PBTRP ;PARITY BYTE TEST TRAP ROUTINE ADDRESS.
1154 001630 002064 .MPPAT: MPPATS ;MEMORY PARITY PATTERN TABLE ADDRESS.
1155 001632 017546 .PESRV: PESRV ;MEMORY PARITY ERROR TRAP ROUTINE ADDRESS.
1156 001634 002314 .EARTB: $ERRTB ;ERROR TYPEOUT TABLE POINTER.
1157 001636 000010 .EIGHT: 8 ;DECIMAL TYPE ROUTINE COUNT DESIGNATOR.
1158 001640 014346 .TST32: TST32 ;SCOPE ABORT ADR FOR WHEN NO MEM AVA FOR TEST.
1159
1160 ;*****
1161 ;* DATA CONTAINERS FOR ERROR PRINTOUT.
1162 ;*****
1162 001642 001116 001120 001124 DT1: $ERRPC,$GDADR,$GDDAT,$BDDAT,0
1163 001650 001126 000000
1164 001654 001512 001116 001120 DT2: $VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT,0
1165 001662 001124 001126 000000
1166 001670 001512 001116 001120 DT12: $VERPC,$ERRPC,$GDADR,$GDDAT,0
1167 001676 001124 000000
1168 001702 001512 001116 001156 DT14: $VERPC,$ERRPC,$TMP0,$GDADR,0
1169 001710 001120 000000
1170 001714 001512 001116 001120 DT15: $VERPC,$ERRPC,$GDADR,$TMP0,$GDDAT,$BDDAT,0
1171 001722 001156 001124 001126
1172 001730 000000
1173 001732 001512 001116 001156 DT21: $VERPC,$ERRPC,$TMP0,$GDADR,$GDDAT,$BDDAT,0
1174 001740 001120 001124 001126
1175 001746 000000
1176 001750 001156 001160 001162 DT22: $TMP0,$TMP1,$TMP2,$TMP3,$GDADR,$GDDAT,$BDDAT,0
1177 001756 001164 001120 001124
1178 001764 001126 000000
1179 001770 001512 001116 001120 DT23: $VERPC,$ERRPC,$GDADR,$BDADR,$GDDAT,$BDDAT,0
1180 001776 001122 001124 001126
1181 002004 000000
1182 002006 001512 001116 001122 DT24: $VERPC,$ERRPC,$BDADR,0
1183 002014 000000
1184 002016 001512 001116 001122 DT25: $VERPC,$ERRPC,$BDADR,$TMP0,$TMP1,0
1185 002024 001156 001160 000000
1186 002032 001512 001116 001156 DT26: $VERPC,$ERRPC,$TMP0,$TMP1,0
1187 002040 001160 000000

```


1188 002044 001156 001160 001120 DT30: \$/MPO,STMP1,\$GDADR,\$BDDAT,0

1189 002052 001126 000000

1190 002056 001164 000000

1191 002062 177777

1192

1193

1194

1195

1196

1197

1198 002064 125325

1199 002066 152652

1200 002070 052452

1201 002072 025125

1202 002074 102070

1203 002076 072527

1204 002100 177777

1205 002102 107030

1206 002104 152525

1207 002106 020000

1208

1209 002110 000000

1210

1211

1212

1213

1214

1215

1216

1217

1218

1219

1220

1221

1222

1223 002112 172101

1224 002114 000000

1225 002116 000000

1226 002120 172103

1227 002122 000000

1228 002124 000000

1229 002126 172105

1230 002130 000000

1231 002132 000000

1232 002134 172107

1233 002136 000000

1234 002140 000000

1235 002142 172111

1236 002144 000000

1237 002146 000000

1238 002150 172113

1239 002152 000000

1240 002154 000000

1241 002156 172115

1242 002160 000000

DT31: STMP3,0
.WORD -1 ;TABLE TERMINATOR.

.SBTTL MEMORY PARITY PATTERNS TABLE

:THE FOLLOWING ARE THE PARITY PATTERNS EXERCISED THRUOUT MEMORY

MPPATS: 125325 :EVEN,ODD
152652 :ODD,EVEN
052452 :EVEN,ODD
025125 :ODD,EVEN
102070 :EVEN,EVEN
072527 :ODD,ODD
177777 :EVEN,EVEN
107030 :ODD,ODD
152525 :ODD,EVEN
0 :EXTRA PATTERN HOLDER FOR
:FUTURE USE
MPENC: 0 ;TABLE TERMINATOR

.SBTTL MEMORY PAPIITY REGISTER ADDRESS TABLE

* THE FOLLOWING REPRESENTS THE MEMORY PARITY REGISTER ADDRESS TABLE
* FROM WHICH PARITY MEMORY IS ADDRESSED & CONTROLLED:
*
* THE LEAST SIGNIFICANT BIT IN THE DEVICE ADDRESS IS SET TO A ONE (1)
* IF THE CONTROL IS FOUND NOT TO BE PRESENT. THE MEMORY PRESENT UNDER
* THE CONTROL OF EACH CONTROLLER IS REPRESENTED BY TWO (2) WORDS FOLLOWING
* THE DEVICE ADDRESS, EACH BIT REPRESENTING A 4K BLOCK. I.E.
* FIRST WORD BIT0 = 0 - 4K, BIT1 = 4 - 8K, ... BIT15 = 60 - 64K
* SECCND WORD BIT0 = 64 - 68K, ... BIT14 = 120 - 124K.

MPRO: 172100 +1 :PARITY STATUS REGISTER
0 :CONTROL MAP (LOW 64K)
0 :CONTROL MAP (HIGH 64K)
MPR1: 172102 +1 :PARITY STATUS REGISTER
0 :CONTROL MAP (LOW 64K)
0 :CONTROL MAP (HIGH 64K)
MPR2: 172104 +1 :PARITY STATUS REGISTER
0 :CONTROL MAP (LOW 64K)
0 :CONTROL MAP (HIGH 64K)
MPR3: 172106 +1 :PARITY STATUS REGISTER
0 :CONTROL MAP (LOW 64K)
0 :CONTROL MAP (HIGH 64K)
MPR4: 172110 +1 :PARITY STAUUS REGISTER
0 :CONTROL MAP (LOW 64K)
0 :CONTROL MAP (HIGH 64K)
MPR5: 172112 +1 :PARITY STATUS REGISTER
0 :CONTROL MAP (LOW 64K)
0 :CONTROL MAP (HIGH 64K)
MPR6: 172114 +1 :PARITY STATUS REGISTER
0 :CONTRCL MAP (LOW 64K)

1244	002162	000000	0		: CONTROL MAP (HIGH 64K)
1245	002164	172117	MPR7: 172116 +1		: PARITY STATUS REGISTER
1246	002166	000000	0		: CONTROL MAP (LOW 64K)
1247	002170	000000	0		: CONTROL MAP (HIGH 64K)
1248	002172	172121	MPR8: 172120 +1		: PARITY STATUS REGISTER
1249	002174	000000	0		: CONTROL MAP (LOW 64K)
1250	002176	000000	0		: CONTROL MAP (HIGH 64K)
1251	002200	172123	MPR9: 172122 +1		: PARITY STATUS REGISTER
1252	002202	000000	0		: CONTROL MAP (LOW 64K)
1253	002204	000000	0		: CONTROL MAP (HIGH 64K)
1254	002206	172125	MPR10: 172124 +1		: PARITY STATUS REGISTER
1255	002210	000000	0		: CONTROL MAP (LOW 64K)
1256	002212	000000	0		: CONTROL MAP (HIGH 64K)
1257	002214	172127	MPR11: 172126 +1		: PARITY STATUS REGISTER
1258	002216	000000	0		: CONTROL MAP (LOW 64K)
1259	002220	000000	0		: CONTROL MAP (HIGH 64K)
1260	002222	172131	MPR12: 172130 +1		: PARITY STATUS REGISTER
1261	002224	000000	0		: CONTROL MAP (LOW 64K)
1262	002226	000000	0		: CONTROL MAP (HIGH 64K)
1263	002230	172133	MPR13: 172132 +1		: PARITY STATUS REGISTER
1264	002232	000000	0		: CONTROL MAP (LOW 64K)
1265	002234	000000	0		: CONTROL MAP (HIGH 64K)
1266	002236	172135	MPR14: 172134 +1		: PARITY STATUS REGISTER
1267	002240	000000	0		: CONTROL MAP (LOW 64K)
1268	002242	000000	0		: CONTROL MAP (HIGH 64K)
1269	002244	172137	MPR15: 172136 +1		: PARITY STATUS REGISTER
1270	002246	000000	0		: CONTROL MAP (LOW 64K)
1271	002250	000000	0		: CONTROL MAP (HIGH 64K)

: THIS IS THE END OF THE TABLE !
 MPRX: .BLKW 17. : TABLE TO HOLD JUST PARITY STATUS REGISTERS THAT EXIST.
 ; (THE EXTRA WORD IS FOR A TERMINATOR.)

.SBTTL ERROR POINTER TABLE

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

::*	EM	:::	POINTS TO THE ERROR MESSAGE
::*	DH	:::	POINTS TO THE DATA HEADER
::*	DT	:::	POINTS TO THE DATA
::*	DF	:::	POINTS TO THE DATA FORMAT

\$ERRTB:
 : * ITEM 1
 DM1 : PARITY REGISTER DATA ERROR.
 DH1 : PC, REG, S/B, WAS
 DT1 : \$ERRPC, \$GDADR, \$GDDAT, \$BDDAT
 DF1 : 16, 18, 16, 16
 : * ITEM 2
 DM2 : ADDRESS TEST ERROR (TST1-5).
 DH2 : V/PC, P/PC, MA, S/B, WAS

1291	002314			
1293	002314	026367		
1294	002316	027771		
1295	002320	001642		
1296	002322	030401		
1298	002324	026423		
1299	002326	030010		

1300	002330	001654	DF2	:SVERPC,SEARPC,SGADR,SGDAT,SBDDAT
1301	002332	030405	DF2	:16,18,18,16,16
1302			:* ITEM 3	
1303	002334	026423	DM2	:ADDRESS TEST ERROR(TST1-5).
1304	002336	030010	DH2	:V/PC,P/PC,MA,S/B,WAS
1305	002340	001654	DT2	:SVERPC,SEARPC,SGADR,SGDAT,SBDDAT
1306	002342	030412	DF3	:16,18,18,8,8
1307			:* ITEM 4	
1308	002344	026457	DM4	:CONSTANT DATA ERROR(TST6-10).
1309	002346	030010	DH2	:V/PC,P/PC,MA,S/B,WAS
1310	002350	001654	DT2	:SVERPC,SEARPC,SGADR,SGDAT,SBDDAT
1311	002352	030405	DF2	:16,18,18,16,16
1312			:* ITEM 5	
1313	002354	026515	DM5	:ROTATING BIT ERROR(TST11-12).
1314	002356	030010	DH2	:V/PC,P/PC,MA,S/B,WAS
1315	002360	001654	DT2	:SVERPC,SEARPC,SGADR,SGDAT,SBDDAT
1316	002362	030405	DF2	:16,18,18,16,16
1317			:* ITEM 6	
1318	002364	026553	DM6	:1 XOR 8 PATTERN ERROR (TST13).
1319	002366	030010	DH2	:V/PC,P/PC,MA,S/B,WAS
1320	002370	001654	DT2	:SVERPC,SEARPC,SGADR,SGDAT,SBDDAT
1321	002372	030405	DF2	:16,18,18,16,16
1322			:* ITEM 7	
1323	002374	026611	DM7	:3 XOR 9 PATTERN ERROR(TST14-17).
1324	002376	030010	DH2	:V/PC,P/PC,MA,S/B,WAS
1325	002400	001654	DT2	:SVERPC,SEARPC,SGADR,SGDAT,SBDDAT
1326	002402	030405	DF2	:16,18,18,16,16
1327			:* ITEM 10	
1328	002404	026652	DM10	:8 XOR 13 PATTERN ERROR(TST20).
1329	002406	030010	DH2	:V/PC,P/PC,MA,S/B,WAS
1330	002410	001654	DT2	:SVERPC,SEARPC,SGADR,SGDAT,SBDDAT
1331	002412	030405	DF2	:16,18,18,16,16
1332			:* ITEM 11	
1333	002414	026711	DM11	:PARITY MEMORY ADDRESS ERROR(TST21).
1334	002416	030010	DH2	:V/PC,P/PC,MA,S/B,WAS
1335	002420	001654	DT2	:SVERPC,SEARPC,SGADR,SGDAT,SBDDAT
1336	002422	030412	DF3	:16,18,18,8,8
1337			:* ITEM 12	
1338	002424	026755	DM12	:DATIP WITH WRONG PARITY DIDN'T TRAP(TST21).
1339	002426	030035	DH12	:V/PC,P/PC,MA,S/B
1340	002430	001670	DT12	:SVERPC,SEARPC,SGADR,SGDAT
1341	002432	030412	DF3	:16,18,18,8
1342			:* ITEM 13	
1343	002434	027031	DM13	:WRONG PARITY TRAPED, BUT NO REGISTER SHOWS ERROR FLAG.
1344	002436	030035	DH12	:V/PC,P/PC,MA,S/B
1345	002440	001670	DT12	:SVERPC,SEARPC,SGADR,SGDAT
1346	002442	030412	DF3	:16,18,18,8
1347			:* ITEM 14	
1348	002444	027121	DM14	:PARITY REGISTER NOT MAPPED AS CONTROLLING THIS ADDRESS(TST21).
1349	002446	030056	DH14	:V/PC,P/PC,REG,MA
1350	002450	001702	DT14	:SVERPC,SEARPC,STMPD,SGADR
1351	002452	030417	DF14	:16,18,18,18
1352			:* ITEM 15	
1353	002454	026367	DM1	:PARITY REGISTER DATA ERROR.
1354	002456	030077	DH15	:V/PC,P/PC,MAUT,REG,S/B,WAS
1355	002460	001714	DT15	:SVERPC,SEARPC,SGADR,STMPD,SGDAT,SBDDAT

1356	002462	030417	DF14	: 16, 18, 18, 18, 16, 16
1357			: * ITEM 16	
1358	002464	027220	DM16	: MORE THAN ONE REGISTER INDICATED PARITY ERROR.
1359	002466	030056	DH14	: V/PC, P/PC, REG, MA
1360	002470	001702	DT14	: \$VERPC, \$ERRPC, \$TMP0, \$GDADR
1361	002472	030417	DF14	: 16, 18, 18, 18
1362			: * ITEM 17	
1363	002474	027277	DM17	: DATA SHOULDN'T HAVE CHANGED WHEN PARITY ERROR
1364				: TRAPPED(TST21).
1365	002476	030010	DH2	: V/PC, P/PC, MA, S/B, WAS
1366	002500	001654	DT2	: \$VERPC, \$ERRPC, \$GDADR, \$GDDAT, \$BDDAT
1367	002502	030412	DF3	: 16, 18, 18, 8, 8
1368			: * ITEM 20	
1369	002504	027375	DM20	: RANDOM DATA ERROR(TST22).
1370	002506	030010	DH2	: V/PC, P/PC, MA, S/B, WAS
1371	002510	001654	DT2	: \$VERPC, \$ERRPC, \$GDADR, \$GDDAT, \$BDDAT
1372	002512	030405	DF2	: 16, 18, 18, 16, 16
1373			: * ITEM 21	
1374	002514	027427	DM21	: INSTRUCTION EXECUTION ERROR(TST23-30).
1375	002516	030132	DH21	: V/PC, P/PC, IUT, MA, S/B, WAS
1376	002520	001732	DT21	: \$VERPC, \$ERRPC, \$TMP0, \$GDADR, \$GDDAT, \$BDDAT
1377	002522	030425	DF21	: 16, 18, 16, 18, 16, 16
1378			: * ITEM 22	
1379	002524	027476	DM22	: "BRANCH GOBBLE" ERROR(TST31).
1380	002526	030163	DH22	: V/PC, P/PC, PS, S/B, PS, MA, S/B, WAS
1381	002530	001750	DT22	: \$TMP0, \$TMP1, \$TMP2, \$TMP3, \$GDADR, \$GDDAT, \$BDDAT
1382	002532	030433	DF22	: 16, 18, 16, 16, 18, 16, 16
1383			: * ITEM 23	
1384	002534	027534	DM23	: PROGRAM CODE CHANGED WHEN RELOCATED.
1385	002536	030226	DH23	: V/PC, P/PC, SRC, MA, DST, MA, S/B, WAS
1386	002540	001770	DT23	: \$VERPC, \$ERRPC, \$GDADR, \$BDADR, \$GDDAT, \$BDDAT
1387	002542	030417	DF14	: 16, 18, 18, 18, 16, 16
1388			: * ITEM 24	
1389	002544	027601	DM24	: TRAPPED, BUT NO REGISTER HAD ERROR BIT SET.
1390	002546	030266	DH24	: V/PC, P/PC, TRP/PC
1391	002550	002006	DT24	: \$VERPC, \$ERRPC, \$BDADR
1392	002552	030417	DF14	: 16, 18, 18
1393			: * ITEM 25	
1394	002554	027655	DM25	: TRAPPED TO 114.
1395	002556	030307	DH25	: V/PC, P/PC, TRP/PC, REG, WAS
1396	002560	002016	DT25	: \$VERPC, \$ERRPC, \$BDADR, \$TMP0, \$TMP1
1397	002562	030417	DF14	: 16, 18, 18, 18, 16
1398			: * ITEM 26	
1399	002564	027675	DM26	: FAILED TO TRAP.
1400	002566	030340	DH26	: V/PC, P/PC, REG, WAS
1401	002570	002032	DT26	: \$VERPC, \$ERRPC, \$TMP0, \$TMP1
1402	002572	030405	DF2	: 16, 18, 18, 16
1403			: * ITEM 27	
1404	002574	027715	DM27	: (ACTION ENABLE WASN'T SET).
1405	002576	030340	DH26	: V/PC, P/PC, REG, WAS
1406	002600	002032	DT26	: \$VERPC, \$ERRPC, \$TMP0, \$BDDAT
1407	002602	030405	DF2	: 16, 18, 18, 16
1408			: * ITEM 30	
1409	002604	000000	0	: NO MESSAGE.
1410	002606	030362	DH30	: REG, WAS, MA, WAS
1411	002610	002044	DT30	: \$TMP0, \$TMP1, \$GDADR, \$BDDAT

K03

MAINDEC-11-DZQMC-B-C: 0-124K MEMORY EXERCISER, 16K VER
DZQMCB.P11 ERROR POINTER TABLE

MACY11 27(732) 10-SEP-76 12:01 PAGE 37

1412 002612 030442
1413
1414 002614 027751
1415 002616 000000
1416 002620 002056
1417 002622 030442

DF30
;* ITEM 31
DM31
0
DT31
DF30

;18,16,18,8
:TRAPPED TO 4
:NO HEADER
:STMP3
:18

1418 .SBTTL START: SETUP AND MAP MEMORY

```

:/*:/*:/*:/*:/*:/*:/*:/*:/*:/*:/*:/*:/*:/*:/*:/*:/*:/*:/*:/*:/*:/*:/*:/*:
: * THIS IS THE NORMAL (SA = 200) BEGINNING OF THE PROGRAM.
: * NOTE: THIS CODE IS NOT POSITION INDEPENDENT.
:/*:/*:/*:/*:/*:/*:/*:/*:/*:/*:/*:/*:/*:/*:/*:/*:/*:/*:/*:/*:/*:/*:/*:/*:
    
```

```

1425 002624 105067 176724 START: CLR B SELFLG ;CLEAR SELECT PARAMETER FLAG.
1425 002630 000403 BR STARTA ;GO DO SETUP AND MEMORY MAP.
1427 002632 112767 177777 176714 SELECT: MOV B #-1, SELFLG ;SET THE SELECT PARAMETERS FLAG.
1428 002640 STARTA:
1429 002640 012706 001100 MOV #SCMTAG,R6 ;:FIRST LOCATION TO BE CLEARED
1430 002644 005026 CLR (R6)+ ;:CLEAR MEMORY LOCATION
1431 002646 022706 001126 CMP #SBODAT,R6 ;:DONE?
1432 002652 001374 BNE .-E ;:LOOP BACK IF NO
1433 002654 012706 001100 MOV #STACK,SP ;:SETUP THE STACK POINTER
1434 002660 012737 000610 000024 MOV #SPWRDN,@PWRVEC ;:POWER FAILURE VECTOR
1435 002666 012737 000340 000026 MOV #340,@PWRVEC+2 ;:LEVEL 7
1436 002674 013746 000004 MOV @#4,-(SP) ;:SAVE ERROR VECTOR
1437 002700 013746 000006 MOV @#6,-(SP)
1438 002704 012767 002720 175072 MOV #64$,4 ;:SET UP TIME OUT VECTOR
1439 002712 005777 176220 TST @SWR ;:TRY TO REFERENCE HARDWARE SWR
1440 002716 000407 BR 65$ ;:BRANCH IF NO TIMEOUT TRAP OCCURS
1441 002720 012767 000176 176210 64$: MOV #SWREG,SWR ;:POINT TO SOFTWARE SWR
1442 002726 012767 000174 176204 MOV #DISPREG,DISPLAY ;:POINT TO SOFTWARE DISPLAY REG
1443 002734 022626 CMP (SP)+,(SP)+ ;:RESTORE STACK
1444 002736 012637 000006 65$: MOV (SP)+,@#6 ;:RESTORE ERROR VECTOR
1445 002742 012637 000004 MOV (SP)+,@#4
1446 002746 005067 176236 CLR $PASS ;:CLEAR PASS COUNT
1447 002752 132767 000200 176243 BIT B #APTSIZE,$ENVM ;:TEST USER SIZE UNDER APT
1448 002760 001403 BEQ 3$ ;:YES,USE NON-APT SWITCH
1449 002762 012767 001224 176146 MOV #SSWREG,SWR ;:NO,USE APT SWITCH REGISTER
1450 002770 3$:
1451 002770 005067 176524 CLR LDDISP ;:CLEAR DISPLAY REGISTER STORAGE LOCN
1452 002774 005077 176140 CLR @DISPLAY ;:CLEAR DISPLAY REGISTER
1453 003000 005227 177777 INC #-1 ;:FIRST TIME?
1454 003004 001017 BNE 66$ ;:BRANCH IF NO
1455 003006 004567 020226 JSR R5 $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
1456 003012 003016 .WORD 67$ ;ADDRESS OF MESSAGE TO BE TYPED
1457 003014 000413 BR 66$ ;GET OVER THE ASCIZ
1458 ;:67$: .ASCIZ <200>'MAINDEC-11-DZQMC-B'<200>
1459 66$:
1460 003044 MOV PC, RO ;GET CURRENT PROGRAM COUNTER.
1461 003046 022700 003046 CMP #, RO ;CHECK IF THE PROGRAM IS RELOCATED.
1462 003052 001402 BEQ 10$ ;BR IF PROGRAM NOT RELOCATED.
1463 003054 000167 175220 JMP RESTAR ;GO TRY TO RELOCTED BEFORE CONTINUING.
1464 003060 012767 000003 175514 10$: MOV #3, PRGMAP ;INITIALIZE PROGRAM MAP....LO 64K.
1465 003066 005067 175512 CLR PRGMAP+2 ;...HI 64K.
1466 003072 005067 175502 CLR RELOCF ;INIT THE RELOCATION FACTOR.
1467
1468 ;* ROUTINE TO SAVE THE LOADERS AT THE END OF BK.
1469 003076 005767 176414 TST LMAD ;CHECK IF LOADERS HAVE BEEN SAVED ALREADY.
1470 003102 001024 BNE 14$ ;BRANCH IF ALREADY SAVED
1471 003104 012700 040000 MOV #40000, RO ;GET END OF BK
1472 003110 010001 MOV RO, R1 ;GET END OF BK
1473 003112 012737 003124 000004 MOV #12$, @ERRVEC ;SET UP TIMEOUT VECTOR
    
```

M03

MAINDEC-11-DZQMC-B-0:
DZQMCB.P11

0-124K MEMORY EXERCISER, 16K VER
START: SETUP AND MAP MEMORY

MACY11 27(732) 10-SEP-76 12:01 PAGE 39

```

1474 003120 011020      11$:  MOV      (R0), (R0)+ ;SEARCH FOR END OF MEMORY
1475 003122 000776      BR      11$ ;KEEP SEARCHING
1475 003124 022626      12$:  CMP      (SP)+, (SP)+ ;RESTORE STACK POINTER
1477 003126 012737 024636 000004  MOV      #ERRTRP, @#ERRVEC ;RESET TIMEOUT VECTOR.
1478 003134 010046      MOV      R0, -(SP) ;SAVE LAST MEMORY ADDRESS (CONTIGUOUS)
1479 003136 012702 002734  MOV      #1500., R2 ;SET UP WORD COUNTER
1480 003142 014041      13$:  MOV      -(R0), -(R1) ;SAVE THE LOADERS
1481 003144 005302      DEC      R2 ;COUNT THE WORDS
1482 003146 001375      BNE     13$ ;BRANCH IF MORE WORDS
1483 003150 012667 176342  MOV      (SP)+, LMAD ;SAVE LAST MEMORY ADDRESS
1484
1485
1486 003154 005067 175426      ;* CHECK IF MEMORY MANAGEMENT IS AVAILABLE, AND SET IT UP IF IT IS.
1487 003160 032777 010000 175750 14$:  CLR      MMAVA ;CLEAR MEM MGMT AVAILABLE FLAG
1488 003166 001014      BIT      #SW12, @SWR ;CHECK FOR INHIBIT KT11 SWITCH
1489 003170 012737 003220 000004  BNE     MAPMEM ;BRANCH IF SET
1490 003176 005037 177572  MOV      #MAPMEM, @#ERRVEC ;SET UP TIMEOUT TRAP VECTOR
1491 003202 004767 011374  CLR      @#SRO ;CLEAR MEM MGMT STATUS REG
1492 003206 005267 175374  JSR     PC, MMINIT ;MEM MGMT INITIALIZATION ROUTINE.
1493 003212 004567 020022  INC     MMAVA ;SET MEM MGMT AVAILABLE FLAG
1494 003216 025076      JSR     R5, $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507 003220 012706 001100  MAPMEM: MOV      #STACK, SP ;RESET THE STACK
1508 003224 012700 001522  MOV      #MEMMAP, R0 ;SET UP MEMORY MAP POINTER...LO 64K.
1509 003230 012701 001524  MOV      #MEMMAP+2, R1 ;...HI 64K.
1510 003234 005010  CLR      (R0) ;CLR MEMORY MAP...LO 64K.
1511 003236 005011  CLR      (R1) ;...HI 64K.
1512 003240 005002  CLR      R2 ;SET ADDRESS POINTER TO 0
1513 003242 012703 000001  MOV      #1, R2 ;SETUP 4K BANK POINTER...LO 64K.
1514 003246 005004  CLR      R4 ;...HI 64K.
1515 003250 005067 175710  CLR      $TMP3 ;INIT TEMPORARY HIGH ADDRESS BITS.
1516 003254 004567 017760  JSR     R5, $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
1517 003260 025143  .WORD   MEMMES ;ADDRESS OF MESSAGE TO BE TYPED
1518
1519 003262 012737 003376 000004  ;"MEMORY MAP:"
1520 003270 011222      1$:  MOV      #2$, @#ERRVEC ;SET UP TIMEOUT VECTOR
1521 003272 032702 017777  MOV      (R2), (R2)+ ;READ+WRITE ALL MEMORY
1522 003276 001374      BIT      #MASK4K, R2 ;CHECK FOR 4K BOUNDRY
1523 003300 050310      BNE     1$ ;BRANCH IF MORE IN BANK
1524 003302 050411      BIS     R3, (R0) ;SET FLAG FOR BANK...LO 64K.
1525 003304 010267 175652      BIS     R4, (R1) ;...HI 64K.
1526 003310 005367 175646      MOV      R2, $TMP2 ;SAVE ADDRESS POINTER.
1527 003314 005767 175266      DEC     $TMP2 ;ADJUST TO LAST ADR, LAST BANK.
1528 003320 001432      TST     MMAVA ;CHECK FOR MEM MGMT.
1529 003322 042767 160000 175632  BEQ     3$ ;BR IF NO MEM MGMT.
1529 003322 042767 160000 175632  BIC     #160000, $TMP2 ;CLEAR BANK BITS ON RELATIVE ADDRESS.

```

N03

MAINDEC-11-DZQMC-B-0: 0-124K MEMORY EXERCISER. 16K VER
 DZQMCB.P11 START: SETUP AND MAP MEMORY

MACY11 27(732) 10-SEP-76 12:01 PAGE 40

```

1530 003330 013705 172344      MOV      Q#KIPAR2,R5      ;SAVE KIPAR2.
1531 003334 005067 175624      CLR      $TMP3          ;MAKE SURE HI BITS ARE INIT.
1532 003340 006305           ASL      R5              ;SHIFT IT 6 PLACES.
1533 003342 006305           ASL      R5
1534 003344 006305           ASL      R5
1535 003346 006305           ASL      R5
1536 003350 006305           ASL      R5
1537 003352 006167 175606      ROL      $TMP3
1538 003356 006305           ASL      R5
1539 003360 006167 175600      ROL      $TMP3
1540 003364 060567 175572      ADD      R5, $TMP2     ;MAKE LAST ADR PHYSICAL.
1541 003370 005567 175570      ADC      $TMP3
1542 003374 000404           BR       3$             ;GO TO UPDATE POINTERS.
1543
1544
1545 003376 022626           ;* TIMEOUT TRAPS TO HERE
2$:  CMP      (SP)+, (SP)+   ;RESTORE THE STACK POINTER
1546 003400 052702 017777      BIS      #MASK4K,R2    ;LAST ADDRESS OF 4K BANK
1547 003404 005202           INC      R2             ;FIRST ADDRESS OF NEXT BANK.
1548 003406 005767 175174      3$:  TST      MMAVA      ;CHECK FOR MEM MGMT
1549 003412 001411           BEQ      4$             ;BRANCH IF NO MEM MGMT
1550 003414 062737 000200 172344      ADD      #200, Q#KIPAR2 ;UPDATE THIRD PAR
1551 003422 012702 040000      MOV      #40000, R2    ;POINT TO START OF THIRD PAR
1552 003426 006303           ASL      R3             ;UPDATE LO BANK POINTER.
1553 003430 006104           ROL      R4             ;UPDATE HI BANK POINTER
1554 003432 100316           BPL      1$             ;BRANCH IF MORE MEMORY TO MAP.
1555 003434 000402           BR       5$             ;EXIT WHEN DONE.
1556
1557 003436 106303           4$:  ASLB     R3           ;UPDATE MAP POINTER
1558 003440 100313           BPL      1$             ;BRANCH IF NOT YET DONE
1559 003442 012737 024636 000004      5$:  MOV      #ERRTRP, Q#ERRVEC ;RESET TIMEOUT VECTOR
1560 003450 004767 015154      JSR      PC, TYPMAP    ;GO TYPE THE MAP.
1561 003454 004567 017560      JSR      R5, $PRINT    ;GO PRINT OUT THE FOLLOWING MESSAGE.
1562 003460 001177           .WORD   $CALF          ;ADDRESS OF MESSAGE TO BE TYPED
1563 003462 011067 176044      MOV      (R0), SAVTST  ;SET UP TEST MAP...LO 64K.
1564 003466 011167 176042      MOV      (R1), SAVTST+2 ;...HI 64K.
1565 003472 011000           MOV      (R0), R0      ;GET LOW MEM MAP
1566 003474 042700 177760      BIC      #177760, R0   ;MASK ALL BUT BOTTOM 4 BANKS
1567 003500 020027 000017      CMP      R0, #17      ;CHECK THAT BOTTOM 16K IS ALL THERE!
1568 003504 001530           BEQ      GMPR          ;BRANCH IF BOTTOM 16K EXISTS
1569 003506 004567 017526      JSR      R5, $PRINT    ;GO PRINT OUT THE FOLLOWING MESSAGE.
1570 003512 025246           .WORD   INSJFF        ;ADDRESS OF MESSAGE TO BE TYPED
1571
1572 003514 000000           6$:  HALT                ;"FIRST 16K OF MEMORY NOT ALL THERE!"
1573
1574
1575
1576
1577
1578 003516 012706 001100      TIMEOUT: MOV #STACK, SP ;SET UP THE STACK POINTER.
1579 003522 005067 175060      CLR      MMAVA        ;CLEAR MEM MGMT AVAILABLE FLAG.
1580 003526 032777 010000 175402      BIT      #SW12, QSWR   ;CHECK IF MEM MGMT TO BE INHIBITED.
1581 003534 001011           BNE      1$           ;BR IF NO MEM MGMT.
1582 003536 012737 003560 000004      MOV      #1$, Q#ERRVEC ;SET TIMEOUT FOR MEM MGMT CHECK.
1583 003544 005037 177572      CLR      Q#SRO        ;CHECK FOR MEM MGMT...TIMES OUT IF NONE.
1584 003550 004767 011026      JSR      PC, MMINIT   ;INIT ALL MEM MGMT REGISTERS.
1585 003554 005267 175026      INC      MMAVA        ;SET MEM MGMT AVAILABLE FLAG.

```


1598	003560		15:	JSR	R5	SPRINT	:GO PRINT OUT THE FOLLOWING MESSAGE.
1599	003560	004567	017454	.WORD	BYTMS		:ADDRESS OF MESSAGE TO BE TYPED
1599	003564	025161					: "BYTE MEMORY MAP."
1599	003566	005000		CLR	R0		:SET UP TYPE OUT FLAG.
1599	003570	005002		CLR	R2		:SET ADDRESS POINTER TO ZERO.
1599	003572	012737	003636	MOV	#200,	2#ERRVEC	:SET TIME OUT VEC TO SERVICE NON-EX MEM.
1599	003600	105712		TSTB	(R2)		:DO DATI ONLY.
1599	003602	032702	000001	BIT	#BIT0,	R2	:CHECK FOR WORD ADDRESS.
1599	003606	001001		BNE	11\$:BR IF ODD BYTE ADDRESS.
1599	003610	011212		MOV	(R2),	(R2)	:DO DATI, DATO...NOP FOR READ ONLY MAP.
1599	003612	151212		BISB	(R2),	(R2)	:DO DATI, DATIP, DATOB...NOP FOR READ ONLY MAP,
1599	003614	005700		TST	R0		:CHECK FOR PREVIOUS TYPED
1599	003616	001023		SNE	30\$:BR IF ALREADY TYPED "FROM".
1600	003620	004567	017414	JSR	R5	SPRINT	:GO PRINT OUT THE FOLLOWING MESSAGE.
1601	003624	025231		.WORD	FROM		:ADDRESS OF MESSAGE TO BE TYPED
1602							: "FROM"
1603	003626	010246		MOV	R2,	-(SP)	:PUT THE DATA ON THE STACK.
1604	003630	004767	021040	JSR	PC,	STYPAD	:DETERMINE THE PHYSICAL ADDRESS AND TYPE IT.
1605	003634	000413		BR	29\$:GO TO ADDRESS POINTER UPDATE.
1606				:* TIME OUTS COME HERE.			
1607	003636	022626		20\$:	CMP	(SP)+,	(SP)+
1608	003640	005700		TST	R0		:POP TWO OFF STACK.
1609	003642	001411		BEG	30\$:CHECK FOR PREVIOUS TYPED
1610	003644	004567	017370	JSR	R5,	SPRINT	:BR IF ALREADY TYPED "TO"
1611	003650	025241		.WORD	TO		:GO PRINT OUT THE FOLLOWING MESSAGE.
1612							:ADDRESS OF MESSAGE TO BE TYPED
1613							: "TO"
1614	003652	005302		DEC	R2		:BACK UP ONE BYTE.
1615	003654	010246		MOV	R2,	-(SP)	:PUT THE DATA ON THE STACK.
1616	003656	004767	021012	JSR	PC,	STYPAD	:DETERMINE THE PHYSICAL ADDRESS AND TYPE IT.
1617	003662	005202		INC	R2		:RESET ADDRESS POINTER.
1618	003664	005100		29\$:	COM	R0	:RESET PREVIOUS TYPED FLAG.
1619	003666	005202		30\$:	INC	R2	:UPDATE ADDRESS POINTER TO NEXT BYTE.
1620	003672	032702	017777	BEG	31\$:EXIT IF ZERO REACHED.
1621	003676	001340		BIT	#MASK4K,R2		:CHECK FOR 4K BANK BOUNDARY.
1622	003700	005767	174702	BNE	10\$:BR IF MORE THIS 4K BANK.
1623	003704	001735		MMAVA			:CHECK IF MEM MGMT IS AVAILABLE.
1624	003706	022737	007600	BEG	10\$:BR IF NO MEM MGMT.
1625	003714	001411		CMP	#7600,	2#KIPAR3	:CHECK FOR END OF LAST 4K BANK.
1626	003716	012702	060000	BEG	31\$:EXIT WHEN ALL DONE.
1627	003722	013737	172346	MOV	#60000,	R2	:RESET VIRTUAL ADDRESS POINTER.
1628	003730	062737	000200	MOV	2#KIPAR3,	2#KIPAR2	:SAVE MEM MGMT REG FOR TYPED.
1629	003736	000720		ADD	#200,	2#KIPAR3	:UPDATE MEM MGMT REG 2 TO NEXT 4K BANK.
1630	003740	005700		BR	10\$:BR BACK TO DO NEXT BANK.
1631	003742	001407		31\$:	TST	R0	:CHECK PREVIOUS TYPE FLAG BEFORE EXIT.
1632	003744	004567	017270	BEG	32\$:BR TO EXIT IF TYPING ALL DONE.
1633	003750	025241		JSR	R5,	SPRINT	:GO PRINT OUT THE FOLLOWING MESSAGE.
1634				.WORD	TO		:ADDRESS OF MESSAGE TO BE TYPED
1635							: "TO"
1636	003752	005302		DEC	R2		:BACK ADDRESS POINTER UP ONE BYTE.
1637	003754	010246	020712	MOV	R2,	-(SP)	:PUT THE DATA ON THE STACK.
1638	003756	004767		JSR	PC,	STYPAD	:DETERMINE THE PHYSICAL ADDRESS AND TYPE IT.
1639	003762	000000		32\$:	HALT		:* THIS ROUTINE IS FOR DEBUG USE ONLY.
1640							:* TO RUN THE MAIN PROGRAM RESTART AT 200 OR 204.
1641	003764	000654		BR	TIMOUT		:LOOP BACK AND DO AGAIN UPON CONTINUE.

```

1642
1643
1644
1645
1646
1647
1648 003766 012704 002252
1649 003772 032777 000100 175136
1650 004000 001035
1651 004002 012703 002112
1652 004006 012737 004030 000004
1653 004014 042713 000001
1654 004020 005773 000000
1655
1656 004024 012324
1657 004026 000403
1658
1659 004030 022626
1660 004032 052723 000001
1661 004036 005023
1662 004040 005023
1663 004042 020327 002252
1664 004046 103762
1665 004050 005014
1666 004052 012737 024636 000004
1667 004060 005767 176166
1668 004064 001006
1669 004066 004567 017146
1670 004072 025327
1671
1672 004074 005014
1673 004076 000167 000544
1674
1675
1676
1677
1678
1679
1680
1681
1682 004102 012703 002252
1683 004106 012302
1684 004110 012700 000001
1685 004114 005012
1686 004116 011201
1687 004120 046701 175370
1688 004124 001405
1689 004126 004767 014376
1690 004132 004767 015700
1691 004136 000001
1692 004140 030067 175350
1693 004144 001025
1694 004146 010012
1695 004150 011201
1696 004152 005012
1697 004154 046701 175334

```

```

.SBTTL MWP PARITY REGISTERS
*****
* SEARCH FOR PARITY REGISTERS PRESENT AND TYPE ADDRESSES OF THOSE FOUND
* THAT ARE FUNCTIONAL AND HAVE CORRESPONDING PARITY MEMORY
*****
GMPR: MOV #MPRX, R4 ;SET UP POINTER TO PARITY REG EXIST TABLE.
      BIT #SW06, JSWR ;CHECK FOR INHIBIT PARITY SWITCH.
      BNE GMPRD ;BR IF INHIBIT PARITY.
      MOV #MPPRD, R3 ;SET UP TABLE POINTER
      MOV #GMPRB, @ERRVEC ;SET UP TIMEOUT TRAP SERVICE
GMPRA: BIC #1, (R3) ;CLEAR FLAG BIT IN TABLE
      TST @R3 ;DOES THIS MEMORY PARITY REGISTER EXIST.
      * IF IT DOESN'T EXIST, A TIMEOUT TRAP WILL GO TO "GMPRB".
      MOV (R3)+, (R4)+ ;SAVE IT IN THE PARITY REG EXIST TABLE.
      BR GMPRC ;SKIP TIMEOUT SERVICE CODE
      * TIMEOUT COMES HERE
GMPRB: CMP (SP)+, (SP)+ ;RESTORE STACK POINTER
      BIS #1, (R3)+ ;SET FLAG TO INDICATE REGISTER NOT PRESENT
GMPRC: CLR (R3)+ ;CLEAR THE MAP...LO 64K.
      CLR (R3)+ ;...HI 64K.
      CMP R3, #MPRX ;HAVE WE CHECKED ALL REGISTERS?
      BLO GMPRA ;NO - GO BACK TO CHECK NEXT ONE
      CLR (R4) ;SET TERMINATOR IN PARITY REG EXIST TABLE.
      MOV #ERRTRP, @ERRVEC ;RESTORE TRAPCATCHER
      TST MPRX ;ANY PARITY REGISTERS PRESENT?
      BNE CTRLS ;YES - GO TEST CONTROLS PRESENT
      JSR R5, SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
      .WORD MTR ;ADDRESS OF MESSAGE TO BE TYPED
      GMPRD: CLR (R4) ;"NO MEMORY PARITY REGISTERS FOUND"
      JMP MANUAL ;MAKE SURE TABLE IS CLEAR.
      ;AND SKIP ALL CONTROLS TESTING

.SBTTL TEST PARITY REGISTERS
*****
* SHOW THAT BITS 0, 2, 5 - 11, AND 15 OF EACH PARITY REGISTER PRESENT
* CAN BE SET AND CLEARED.
* THIS IS A CNCE ONLY TEST.
*****
CTRLS: MOV #MPRX, R3 ;LOAD INITIAL TABLE ADDRESS FOR A POINTER
15: MOV (R3)+, R2 ;LOAD R2 WITH ADDRESS OF THIS PARITY REGISTER
      MOV #1, R0 ;LOAD R0 WITH VALUE OF 1ST BIT TESTED
      CLR (R2) ;INITIALIZE THE PARITY REGISTER
      MOV (R2), R1 ;READ THE CONTENTS OF THE PARITY REGISTER
      BIC RESRVD, R1 ;CLEAR BITS WHICH ARE RESERVED
      BEQ 25 ;CHECK OTHER BITS - BRANCH IF JK
645: JSR PC, SPRINT ;SET UP VALUES FOR ERROR PRINTING.
      JSR PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)
      .WORD 1 ;ERROR TYPE CODE.
25: BIT R0, RESRVD ;IS THIS BIT RESERVED?
      BNE 35 ;YES - DON'T TEST IT
      MOV R0, (R2) ;NO - SET THIS BIT IN THE PARITY REGISTER
      MOV (R2), R1 ;READ & SAVE CONTENTS OF THE PARITY REGISTER
      CLR (R2) ;CLEAR THE PARITY REGISTER
      BIC RESRVD, R1 ;CLEAR BIT LOCATIONS THAT ARE RESERVED

```

```

1698
1699
1700 004160 020001
1701 004162 001405
1702 004164 004767 014370
1703 004170 004767 015642
1704 004174 000001
1705 004176
1706 004176 011201
1707 004200 046701 175310
1708 004204 001405
1709 004206 004767 014316
1710 004212 004767 015620
1711 004216 000001
1712 004220 006300
1713 004222 103346
1714 004224 005713
1715 004226 001327
1716
1717
1718
1719
1720
1721
1722 004230 012703 002252
1723 004234 012733 107745
1724 004240 005713
1725 004242 001374
1726 004244 000005
1727 004246 012703 002252
1728 004252 012302
1729 004254 011201
1730 004256 005012
1731 004260 042701 077772
1732 004264 005701
1733 004266 001405
1734 004270 004767 014224
1735 004274 004767 015536
1736 004300 000001
1737 004302
1738 004302 005713
1739 004304 001362
  
```

: IMPORTANT NOTE:

```

:
:      CMP      R0,
:      BEQ      66$
65$:   JSR      PC,
:      JSR      PC,
:      .WORD    1
56$:   MOV      (R2), R1
:      BIC      RESRVD, R1
:      BEQ      3$
57$:   JSR      PC,
:      JSR      PC,
:      .WORD    1
3$:    ASL      R0
:      BCC      2$
:      TST      (R3)
:      BNE      1$
  
```

```

THE FOLLOWING TEST WILL FAIL ON A PDP-11/45 WITH
MOS PARITY REGISTERS. PATCH "RESRVD"(1514) TO 77772.
R1      :COMPARE THE CHECK WORD WITH THE DATA READ.
:      :BRANCH OVER ERROR CALL IF GOOD DATA.
SPRNT0  :SET UP VALUES FOR ERROR PRINTING.
$ERROR  :*** ERROR *** (GO TYPE A MESSAGE)
:      :ERROR TYPE CODE.

:READ THE CONTENTS OF THE PARITY REGISTER
:CLEAR BITS WHICH ARE RESERVED
:CHECK OTHER BITS - BRANCH IF OK
SPRNT   :SET UP VALUES FOR ERROR PRINTING.
$ERROR  :*** ERROR *** (GO TYPE A MESSAGE)
:      :ERROR TYPE CODE.
:ROTATE TO GET NEXT BIT TO BE TESTED
:BRANCH IF NOT DONE WITH ALL BITS
:CHECK FOR TABLE TERMINATOR.
:BR IF MORE PARITY REGISTERS.
  
```

```

*****
* SHOW THAT RESET CLEARS BITS 0,2, AND 15 OF EACH PARITY REGISTER PRESENT.
* THIS IS A ONCE ONLY TEST.
*****
  
```

```

RESCHK: MOV      #MPRX, R3      :LOAD INITIAL TABLE ADDRESS FOR A POINTER
1$:     MOV      #107745,3(R3)+ :SET ALL DEFINED BITS TO A ONE (1)
:      TST      (R3)           :CHECK FOR TABLE TERMINATOR.
:      BNE      1$           :BRANCH IF MORE REGISTERS.
:      RESET                    :ISSUE AN INIT TO THE WORLD.
2$:     MOV      #MPRX, R3      :LOAD INITIAL TABLE ADDRESS FOR A POINTER
:      MOV      (R3)+, R2      :STORE THE PARITY REGISTER ADDRESS
:      MOV      (R2), R1       :GET CONTENTS OF REGISTER
:      CLR      (R2)           :MAKE SURE IT IS CLEAR.
:      BIC      #77772, R1     :MASK THE BITS RESERVED FOR FOR FUTURE USE
:      TST      R1             :CHECK IF REST WERE CLEARED BY THE RESET
:      BEQ      65$           :BRANCH OVER ERROR CALL IF GOOD DATA.
64$:   JSR      PC, SPRNT      :SET UP VALUES FOR ERROR PRINTING.
:      JSR      PC, $ERROR     :*** ERROR *** (GO TYPE A MESSAGE)
:      .WORD    1             :ERROR TYPE CODE.
65$:   TST      (R3)           :CHECK FOR TABLE TERMINATOR.
:      BNE      2$           :BR IF MORE PARITY REGISTERS.
  
```

1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795

004306 004767 014174
004312 012767 000001 175222
004320 005067 175220
004324 005002
004326 005767 174254
004332 001404
004334 012702 040000
004340 004767 010236

004344 005067 175166
004350 005067 175164
004354 012703 002112
004360 032713 000001
004364 001027
004366 012773 000004 000000

004374 011212
004376 005712
004400 042773 000004 000000
004406 005773 000000

004412 100014

004414 056763 175122 000002
004422 056763 175116 000004
004430 056767 175106 175100
004436 056767 175102 175074
004444 062703 000006 38:
004450 020327 002252
004454 103741
004456 011212
004460 005767 174122
004464 001425
004466 062737 000200 172344 43:
004474 006367 175042
004500 006167 175040
004504 100422

```
.SBTTL MWP PARITY MEMORY
*****
MAP CORRESPONDENCE BETWEEN PARITY REGISTERS AND MEMOP, AND TYPE RESULTS
NOTE THAT IF PARITY MEMORY IS NOT LOCATED CORRECTLY THAT IT IS IN ALL
PROBABILITY DUE TO ONE OF THE FOLLOWING FAILURES:
- SETTING WRITE WRONG PARITY DIDN'T CAUSE BAD PARITY TO BE WRITTEN
- PARITY GENERATE OR DETECT LOGIC FAILED
- PARITY ERROR BIT FAILED TO SET
- PARITY BITS IN MEMORY LOCATION FAILED
- I.E. BIT STUCK AT GOOD PARITY VALUE
*****

JSR PC, CLRPAR ;INITIALIZE ALL PARITY REGISTERS
MOV #1, BITPT ;INITIALIZE 4K POINTER
CLR BITPT+2 ;CLEAR HI 64K POINTER
CLR R2 ;SET ADR POINTER TO ZERO
TST MMAVA ;CHECK FOR MEM MGMT
BEQ MAPRB ;BRANCH IF NO MEM MGMT
MOV #40000, R2 ;SET ADR POINTER TO PAR2
JSR PC, MMINIT ;SET UP ALL MEMORY MGMT REGISTERS.

*****
;SET WRITE WRONG PARITY IN ALL REGISTERS PRESENT
;* THEN WRITE TEST LOCATION VIA DAT0 & READ TEST LOCATION VIA DAT1
;* THEN CLEAR WRITE WRONG PARITY IN ALL REGISTERS.
*****

MAPRB: CLR PMEMAP ;CLEAR THE PARITY MEMORY MAP
CLR PMEMAP+2
18: MOV #MPRO, R3 ;INITIALIZE TABLE ADDRESS
28: BIT #1, (R3) ;IS THIS REGISTER PRESENT?
BNE 38 ;NO - GET THE NEXT ONE
MOV #WWP, 2(R3) ;YES - SET WRITE WRONG PARITY
;AND CLEAR REST OF REGISTER
MOV (R2), (R2) ;WRITE WRONG PARITY
TST (R2) ;READ WRONG PARITY
BIC #WWP, 2(R3) ;CLEAR WRITE WRONG PARITY
TST 2(R3) ;OTHERWISE, CHECK TO SEE IF THIS
;CONTROL REGISTER GOT A PARITY
;ERROR
BPL 38 ;BRANCH IF IT DIDN'T AND CHECK
;NEXT POSSIBLE ONE
BIS BITPT, 2(R3) ;SET FLAG IN MAP FOR THIS PARITY REGISTER
BIS BITPT+2, 4(R3)
BIS BITPT, PMEMAP ;SET FLAG IN PARITY MAP
BIS BITPT+2, PMEMAP+2
38: ADD #6, R3 ;STEP UP TO NEXT REGISTER
CMP R3, #MPRX ;ARE WE DONE WITH TABLE?
BLO 28 ;GO BACK TO CHECK FOR ANY MORE!
MOV (R2), (R2) ;CLEAR BAD PARITY
TST MMAVA ;CHECK FOR MEM MGMT
BEQ 108 ;BR IF NO MEM MGMT
43: ADD #200, 2#KIPAR2 ;UPDATE PAR TO NEXT 4K BANK.
ASL BITPT ;UPDATE BANK POINTER...LO 64K.
ROL BITPT+2 ;...HI 64K.
BMI TMAP ;BR IF ALL DONE.
```

```

1796 004506 036767 175030 175006
1797 004514 001317
1798 004516 036767 175022 175000
1799 004524 001313
1800 004526 000757
1801 004530 036767 175006 174764 115:
1802 004536 001306
1803 004540 062702 020000 105:
1804 004544 106367 174772
1805 004550 100367
1806
1807
1808
1809
1810
1811
1812 004552 004767 013730
1813 004556 004567 016456
1814 004562 025204
1815
1816 004564 012703 002112
1817 004570 032713 000001
1818 004574 001017
1819 004576 004567 016436
1820 004602 025613
1821
1822 004604 011346
1823
1824
1825 004606 013746 177776
1826 004612 004767 017614
1827 004616 004567 016416
1828 004622 025632
1829
1830 004624 010300
1831 004626 005720
1832 004630 004767 013774
1833 004634 062703 000006
1834 004640 020327 002252
1835 004644 103751
1836
1837 004646 012700 000014
1838 004652 012701 001560
1839 004656 005021
1840 004660 005300
1841 004662 001375
1842 004664 105767 174664
1843 004670 001005
1844 004672 016767 174264 174672
1845 004700 000167 000402

```

```

BIT BITPT, MEMMAP ;CHECK IF BANK EXISTS...LO 64K.
BNE 15 ;BR IF BANK EXISTS.
BIT BITPT+2, MEMMAP+2 ;...HI 64K.
BNE 15 ;BR IF BANK EXISTS.
BR 45 ;BR IF BANK DOESN'T EXIST.
BIT BITPT, MEMMAP ;CHECK IF BANK EXISTS.
BNE 15 ;BR IF BANK EXISTS.
ADD #20000, R2 ;UPDATE ADDRESS POINTER TO NEXT BANK.
ASLB BITPT ;MOVE POINTER TO NEXT BANK.
BPL 115 ;BR IF MORE TO LOOK FOR.

*****
* ROUTINE TO TYPE MAP OF WHERE PARITY MEMORY IS PRESENT
* AND WHICH CONTROL REGISTERS CONTROL WHICH MEMORY
*****

TMAP: JSR PC, CLRPAR ;INITIALIZE ALL PARITY REGISTERS PRESENT
      JSR R5, $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
      .WORD MTMAP ;ADDRESS OF MESSAGE TO BE TYPED
      ;"PARITY MEMORY MAP:"
      MOV #MPRO, R3 ;INITIALIZE TABLE POINTER
      BIT #BIT0, (R3) ;CHECK IF THIS REGISTER IS PRESENT.
      BNE 25 ;BR IF NOT PRESENT.
      JSR R5, $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
      .WORD MX1 ;ADDRESS OF MESSAGE TO BE TYPED
      ;"REGISTER AT"
      MOV (R3), -(SP) ;SAVE (R3) FOR TYPEOUT
      ;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOC ROUTINE
      ;* WITHOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
      MOV #PSW, -(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
      JSR PC, $TYPOC ;GO TO THE SUBROUTINE
      JSR R5, $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
      .WORD MX2 ;ADDRESS OF MESSAGE TO BE TYPED
      ;"CONTROLS"
      MOV R3, R0 ;SET UP R0 FOR TYPMAP ROUTINE.
      TST (R0)+ ;UPDATE POINTER TO MAP.
      JSR PC, TYPMAP ;GO TYPE THE MEMORY COVERED BY THIS REGISTER.
      ADD #6, R3 ;UPDATE TO NEXT REGISTER IN TABLE.
      CMP R3, #MPRX ;ARE WE ALL DONE WITH TABLE?
      BLO 15 ;BRANCH IF MORE REGISTERS

MANJAL: MOV #12, R0 ;SET COUNTER TO CLEAR 12 WORDS.
        MOV #FSTADR, R1 ;STARTING AT FSTADR.
        CLR (R1)+ ;CLEAR THE LOCATIONS.
        DEC R0 ;COUNT.
        BNE 15 ;BR IF MORE.
        TSTB SELFLG ;CHECK FOR SELECT PARAMETERS STARTUP.
        BNE MANUL1 ;BR IF PARAMETERS TO BE SELECTED.
        MOV $TMP2, LSTADR ;SET UP VIRTUAL LAST ADDRESS.
        JMP MANUL2 ;SKIP PARAMETER SELECTION SECTION.

```

Handwritten mark resembling a stylized 'd' or '4'.

```

1846
1847
1848
1849
1850 004704 012700 000001
1851 004710 005001
1852 004712 005002
1853 004714 005003
1854 004716 004567 016316
1855 004722 025570
1856
1857
1858
1859 004724 013746 177776
1860 004730 004767 016132
1861 004734 042716 000001
1862 004740 005067 174566
1863 004744 005067 174564
1864 004750 062702 020000
1865 004754 005503
1866 004756 020367 016254
1867 004762 103403
1868 004764 101006
1869 004766 020216
1870 004770 101004
1871 004772 006300
1872 004774 006101
1873 004776 100364
1874 005000 000507
1875 005002 030567 174514
1876 005006 001003
1877 005010 030167 174510
1878 005014 001501
1879 005016 016704 016214
1880 005022
1881 005022 004567 016212
1882 005026 025755
1883
1884
1885
1886 005030 013746 177776
1887 005034 004767 016026
1888 005040 005716
1889 005042 001010
1890 005044 005767 016166
1891 005050 001025
1892 005052 016716 174104
1893 005056 016767 174102 016152
1894 005064 012667 174502
1895 005070 020467 016142
1896 005074 101352
1897 005076 103403
1898 005100 021667 174466
1899 005104 101346
1900 005106 032716 017777
1901 005112 001404

```

```

.SBTTL USER PARAMETER SELECTION SECTION
*****
;* USER PARAMETER SELECTION SECTION IS ENTERED BY STARTING AT 204.
*****
MANUL!: MOV #BIT0, R0 ;SET UP BANK POINTER.
        CLR R1 ;...HI 64K.
        CLR R2 ;CLEAR ADDRESS POINTER.
        CLR R3 ;...HI ADDRESS BITS.
        JSR R5, $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
        .WORD FADMES ;ADDRESS OF MESSAGE TO BE TYPED
        ;"FIRST ADDRESS:"
;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $RDOCT ROUTINE
;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSTEMAC**.
        MOV @#PSW, -(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
        JSR PC, $RDOCT ;GO TO THE SUBROUTINE
        BIC #BIT0, (SP) ;MAKE SURE ADDRESS IS ON A WORD BOUNDARY.
        CLR SAVTST ;INIT TEST MAP...LO 64K.
        CLR SAVTST+2 ;...HI 64K.
1$: ADD #20000, R2 ;UPDATE ADDRESS POINTER TO NEXT BANK.
        ADC R3
        CMP R3, $HI0CT ;CHECK HI ADDRESS BITS.
        BLO 2$ ;BR IF NOT HI ENOUGH YET.
        BHI 3$ ;BR IF PAST SELECTED ADDRESS.
        CMP R2, (SP) ;CHECK THE LO ADDRESS BITS.
        BHI 3$ ;BR IF PAST SELECTED ADDRESS.
2$: ASL R0 ;UPDATE POINTER...LO 64K.
        ROL R1 ;...HI 64K.
        BPL 1$ ;BR BACK TO CHECK NEXT BANK.
        BR 17$ ;BR IF OVERFLOW.
3$: BIT R0, MEMMAP ;CHECK IF BANK EXISTS.
        BNE 4$ ;BR IF BANK EXISTS.
        BIT R1, MEMMAP+2 ;CHECK HI 64K.
        BEQ 17$ ;BR IF ADDRESS IN UN-MAPPED BANK.
4$: MOV $HI0CT, R4 ;SAVE FIRST ADR HI BITS.
10$: JSR R5, $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
        .WORD LADMES ;ADDRESS OF MESSAGE TO BE TYPED
        ;"LAST ADDRESS:"
;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $RDOCT ROUTINE
;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSTEMAC**.
        MOV @#PSW, -(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
        JSR PC, $RDOCT ;GO TO THE SUBROUTINE
        TST (SP) ;CHECK IF ADR 0 SELECTED (DEFAULT).
        BNE 11$ ;BR IF NOT 0 (DEFAULT)
        TST $HI0CT ;CHECK HI BITS.
        BNE 11$ ;BR IF NOT 0 (DEFAULT).
        MOV $TMP2, (SP) ;SET UP DEFAULT LAST ADR.
        MOV $TMP3, $HI0CT
11$: MOV (SP)+, LSTADR ;GET THE DATA.
        CMP R4, $HI0CT ;CHECK FOR LAST ADR BELOW FIRST ADR.
        BHI 10$ ;BR IF LAST BELOW FIRST.
        BLO 12$ ;BR IF LAST ABOVE FIRST.
        CMP (SP), LSTADR ;CHECK FOR LAST BELOW FIRST.
        BHI 10$ ;BR IF LAST BELOW FIRST.
12$: BIT #MASK4K, (SP) ;CHECK IF FIRST ADR ON BANK BOUNDARY.
        BEQ 13$ ;BR IF ON BOUNDARY.

```

Handwritten mark resembling a stylized '4'.

```

:902 005114 J10067 174446      MOV    R0,    FADMAP ;SET UP FIRST ADDRESS MAP.
:903 005120 010167 174444      MOV    R1,    FADMAP+2
:904 005124 050067 174402      13$:  BIS    R0,    SAVTST ;SET FLAG IN TEST MAP...LO 64K.
:905 005130 050167 174400      BIS    R1,    SAVTST+2 ;...HI 64K.
:906 005134 020367 016076      14$:  CMP    R3,    $HI0CT ;CHECK FOR PAST LAST ADR.
:907 005140 103404                BLO    15$ ;BR IF BELOW LAST ADR.
:908 005142 101020                BHI    16$ ;BR IF GONE PAST LAST ADR.
:909 005144 020267 174422      CMP    R2,    LSTADR ;CHECK FOR PAST LAST ADR.
:910 005150 101015                BHI    16$ ;BR IF GONE PAST LAST ADR.
:911 005152 062702 020000      15$:  ADD    #20000, R2 ;UPDATE ADDRESS POINTER.
:912 005156 005503                ADC    R3,    ;...HI BITS.
:913 005160 005300                ASL    R0,    ;UPDATE BANK POINTER...LO 64K.
:914 005162 006101                ROL    R1,    ;...HI 64K.
:915 005164 100415                BMI    17$ ;BR IF OVERFLOW.
:916 005166 030067 174330      BIT    R0,    MEMMAP ;CHECK IF THIS BANK EXISTS.
:917 005172 001354                BNE    13$ ;BR IF BANK EXISTS.
:918 005174 030167 174324      BIT    R1,    MEMMAP+2 ;CHECK IF THIS BANK EXISTS.
:919 005200 001351                BNE    13$ ;BR IF BANK EXISTS.
:920 005202 000754                BR     14$ ;BR IF BANK DOESN'T EXIST.
:921 005204 030067 174312      16$:  BIT    R0,    MEMMAP ;CHECK IF THIS BANK EXISTS.
:922 005210 001010                BNE    20$ ;BR IF IT EXISTS.
:923 005212 030167 174306      BIT    R1,    MEMMAP+2 ;CHECK IF THIS BANK EXISTS.
:924 005216 001005                BNE    20$ ;BR IF IT EXISTS.
:925 005220 005726      17$:  TST    (SP)+ ;ADJUST THE STACK.
:926 005222 004567 016012      JSR    R5,    $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
:927 005226 026000                .WORD  BADADR ;ADDRESS OF MESSAGE TO BE TYPED
:928                                ;"?ADDRESS IN UNMAPPED BANK?"
:929 005230 000606                BR     MANUAL ;LOOP BACK TO THE BEGINNING.
:930 005232 010067 174342      20$:  MOV    R0,    LADMAP ;SET UP MAP FOR LAST ADDRESS.
:931 005236 010167 174340      MOV    R1,    LADMAP+2
:932 005242 005767 173340      21$:  TST    MMAPVA ;CHECK FOR MEMORY MANAGEMENT.
:933 005246 001404                BEQ    22$ ;BR IF NO MEM MGMT.
:934 005250 042716 160000      BIC    #160000, (SP) ;ADJUST FSTADR TO VIRTUAL BANK C.
:935 005254 062716 040000      ADD    #40000, (SP) ;...TO VIRTUAL BANK 2.
:936 005260 012667 174274      22$:  MOV    (SP)+, FSTADR ;SAVE FIRST ADDRESS OFF THE STACK.
:937 005264                30$:
:938 005264 004567 015750      JSR    R5,    $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
:939 005270 026035                .WORD  CONST ;ADDRESS OF MESSAGE TO BE TYPED
:940                                ;"SELECT CONSTANT:"
:941                                ;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $RDOCT ROUTINE
:942                                ;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSTEMAC**.
:943 005272 013746 177776      MOV    @#PSW, -(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
:944 005276 004767 015564      JSR    PC,    $RDOCT ;GO TO THE SUBROUTINE
:945 005302 012667 174300      MOV    (SP)+, .CONST ;SAVE THE CONSTANT
:946 005306 005767 173274      MANUL2: TST    MMAPVA ;CHECK IF MEM MGMT IS AVAILABLE.
:947 005312 001406                BEQ    31$ ;BR IF NO MEM MGMT.
:948 005314 042767 160000 174250      BIC    #160000, LSTADR ;ADJUST LSTADR TO VIRTUAL BANK C.
:949 005322 062767 040000 174242      ADD    #40000, LSTADR ;...VIRTUAL BANK 2.
:950 005330 062767 000002 174234      31$:  ADD    #2,    LSTADR ;ADJUST LAST ADDRESS UP ONE WORD.
:951 005336 042767 000001 174226      BIC    #BIT0, LSTADR ;MAKE SURE IT IS A WORD ADDRESS.
:952 005344 032767 017777 174220      BIT    #MASK4K, LSTADR ;CHECK IF LAST ADR IS ON BANK BOUNDARY.
:953 005352 001004                BNE    START1 ;BR IF NOT ON BOUNDARY.
:954 005354 005067 174220      CLR    LADMAP ;CLEAR OUT THE LAST ADDRESS MAP.
:955 005360 005067 174216      CLR    LADMAP+2
:956

```

111
65

MAINDEC-11-DZQMC-B-C: 0-124K MEMORY EXERCISER, 16K VER
DZQMCB.P11 USER PARAMETER SELECTION SECTION

*** THE REST OF THE PROGRAM IS POSITION INDEPENDENT CODE, SO THAT IT CAN EXECUTE PROPERLY WHEN THE PROGRAM HAS BEEN RELO
* THIS IS DONE SO THAT THE FIRST TWO BANKS OF MEMORY CAN BE EXERCISED IN EXACTLY THE SAME MANNER AS THE REST OF MEMORY

1963	005364	016706	174220		START1: MOV	.STACK	SP	:SET STACK POINTER
1964	005370	012767	005364	173510	MOV	#START1.	\$LPADR	:INIT LOOP ADDRESS.
1965	005376	066767	173176	173502	ADD	RELOCF.	\$LPADR	
1966	005404	004767	012344		JSR	PC	MAMF	:SET UP MEMORY PARITY ERROR VECTOR
1967	005410	005767	173172		TST	MMAVA		:CHECK FOR MEMORY MANAGEMENT AVAILABLE.
1968	005414	001406			BEQ	TST1		:BRANCH IF NO MEM MGMT.
1969	005416	032737	000001	177572	BIT	#BIT0.	Q#SRC	:CHECK IF MEM MGMT ENABLED.
1970	005424	001002			SNE	TST1		:BR IF MEM MGMT ENABLED.
1971	005426	004767	007150		JSR	PC,	MMINIT	:SET UP MEM MGMT REGISTERS.


```

1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982 005432
1983 005432 004567 013410
1984 005436 000001
1985
1986 005440 000167 006702
1987
1988 005444 012767 000001 173534
1989
1990 005452 004467 007252
1991 005456 004767 010674
1992 005462 010012
1993 005464 012201
1994 005466 020001
1995 005470 001405
1996 005472 004767 013106
1997 005476 004767 014334
1998 005502 000002
1999 005504
2000 005504 062700 000002
2001 005510 030502
2002 005512 001363
2003 005514 004767 007766
2004
2005
2006
2007 005520 004467 007642
2008 005524 004767 010626
2009 005530 162700 000002
2010 005534 014201
2011 005536 020001
2012 005540 001405
2013 005542 004767 013012
2014 005546 004767 014264
2015 005552 000002
2016 005554
2017 005554 030502
2018 005556 001364
2019 005560 004767 010412

```

```

.SBTTL SECTION 1: MEMORY ADDRESS TESTS
*****
*TEST 1 WRITE VALUE OF MEMORY ADDRESS INTO MEMORY
* RO = DATA WRITTEN INTO MEMORY (SHOULD BE)
* RI = DATA READ FROM MEMORY (WAS)
* R2 = VIRTUAL ADDRESS
* R3 = NOT USED
* R4 = NOT USED
* R5 = BLOCK BOUNDARY BIT MASK.
*****
†ST1:
      JSR      R5,      $SCOPE ;GO TO SCOPE ROUTINE.
      .WORD   1          ;MINIMUM BLOCK SIZE OF 1 WORDS
                          ;REQUIRED FOR THIS TEST.
      JMP     TST32      ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                          ;AVAILABLE FOR TEST.
                          ;;SET TEST NUMBER IN MAIL BOX
* UPWARDS WORD ADDRESSING.
      MOV     #STN-1,$STEN
      JSR     R4,      INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$:   JSR     PC,      PHYADR ;GET PHYSICAL ADDRESS INTO R0
2$:   MOV     R0,      (R2)   ;WRITE VALUE OF ADDRESS INTO ADDRESS
      MOV     (R2)+,   R1     ;GET THE DATA FROM MEMORY UNDER TEST.
      CMP     R0,      R1     ;COMPARE THE CHECK WORD WITH THE DATA READ.
      BEQ    65$      ;BRANCH OVER ERROR CALL IF GOOD DATA.
64$:  JSR     PC,      SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
      JSR     PC,      $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
      .WORD   2          ;ERROR TYPE CODE.
65$:  ADD     #2,      R0     ;ADD #2 TO PHYSICAL ADDRESS
      BIT     R5,      R2     ;CHECK FOR END OF A BLOCK.
      BNE    2$      ;BRANCH IF MORE IN CURRENT BLOCK.
      JSR     PC,      MMUP   ;FIND NEXT BLOCK AND LOOP TO 1$.
* CHECK THAT VALUE OF MEMORY ADDRESS WAS WRITTEN CORRECTLY
* DOWNWARDS WORD ADDRESSING.
      JSR     R4,      INITDN ;INITIALIZE THE MEMORY ADDRESS POINTERS.
3$:   JSR     PC,      PHYADR ;GET PHYSICAL ADDRESS INTO R0
4$:   SUB     #2,      R0     ;DEC DATA BY 2
      MOV     -(R2),   R1     ;GET THE DATA FROM MEMORY
      CMP     R0,      R1     ;COMPARE THE CHECK WORD WITH THE DATA READ.
      BEQ    67$      ;BRANCH OVER ERROR CALL IF GOOD DATA.
66$:  JSR     PC,      SPRNT0 ;SET UP VALUES FOR ERROR PRINTING.
      JSR     PC,      $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
      .WORD   2          ;ERROR TYPE CODE.
67$:  BIT     R5,      R2     ;CHECK FOR END OF A BLOCK.
      BNE    4$      ;BRANCH IF MORE IN CURRENT BLOCK.
      JSR     PC,      MMDOWN ;FIND NEXT BLOCK AND LOOP TO $TAG1.

```

```

2020
2021
2022
2023
2024
2025
2026
2027
2028
2029 005564
2030 005564 004567 013256
2031 005570 000000
2032 005572 012767 000002 173406
2033
2034 005600 004467 007124
2035 005604 004767 010546
2036 005610 110022
2037 005612 005200
2038 005614 030502
2039 005616 001374
2040 005620 004767 007662
2041
2042
2043
2044 005624 004467 007536
2045 005630 004767 010522
2046 005634 005300
2047 005636 114201
2048 005640 120001
2049 005642 001405
2050 005644 004767 012710
2051 005650 004767 014162
2052 005654 000003
2053 005656
2054 005656 030502
2055 005660 001365
2056 005662 004767 010310
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067 005666
2068 005666 004567 013154
2069 005672 000000
2070 005674 012767 000003 173304
2071
2072 005702 004467 007460
2073 005706 004767 010444
2074 005712 005100
2075 005714 062700 000002

```

```

;*****
;*TEST 2 WRITE VALUE OF MEMORY ADDRESS INTO MEMORY
;* R0 = DATA WRITTEN INTO MEMORY (SHOULD BE)
;* R1 = DATA READ FROM MEMORY (WAS)
;* R2 = VIRTUAL ADDRESS
;* R3 = NOT USED
;* R4 = NOT USED
;* R5 = BLOCK BOUNDARY BIT MASK.
;*****
†ST2:
JSR R5, $SCOPE ;GO TO SCOPE ROUTINE.
.WORD 0 ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
MOV #STN-1,$TESTN ;;SET TEST NUMBER IN MAIL BOX
;* UPWARDS BYTE ADDRESSING.
JSR R4, INITM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$: JSR PC, PHYADR ;GET PHYSICAL ADDRESS INTO R0
2$: MOVB RO, (R2)+ ;WRITE VALUE OF ADDRESS INTO ADDRESS
INC RO ;ADD ONE TO PHYSICAL ADDRESS
BIT R5, R2 ;CHECK FOR END OF A BLOCK.
BNE 2$ ;BRANCH IF MORE IN CURRENT BLOCK.
JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 1$.

;* CHECK THAT VALUE OF MEMORY ADDRESS WAS WRITTEN CORRECTLY
;* DOWNWARDS BYTE ADDRESSING.
JSR R4, INITM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
3$: JSR PC, PHYADR ;GET PHYSICAL ADDRESS INTO R0
4$: DEC RO ;DEC DATA BY 1
MOVB -(R2), R1 ;GET THE DATA FROM MEMORY
CMPB RO, R1 ;CHECK THE DATA...LO BYTE ONLY VALID.
BEQ 65$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
64$: JSR PC, SPRNTO ;SET UP VALUES FOR ERROR PRINTING.
JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
.WORD 3 ;ERROR TYPE CODE.
65$: BIT R5, R2 ;CHECK FOR END OF A BLOCK.
BNE 4$ ;BRANCH IF MORE IN CURRENT BLOCK.
JSR PC, MMDOWN ;FIND NEXT BLOCK AND LOOP TO $TAG1.

;*****
;*TEST 3 WRITE 1'S COMPLEMENT VALUE OF ADDRESS INTO ADDRESS.
;* R0 = DATA WRITTEN INTO MEMORY (SHOULD BE)
;* R1 = DATA READ FROM MEMORY (WAS)
;* R2 = VIRTUAL ADDRESS
;* R3 = NOT USED
;* R4 = NOT USED
;* R5 = BLOCK BOUNDARY BIT MASK.
;*****
†ST3:
JSR R5, $SCOPE ;GO TO SCOPE ROUTINE.
.WORD 0 ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
MOV #STN-1,$TESTN ;;SET TEST NUMBER IN MAIL BOX
;* DOWNWARDS WORD ADDRESSING.
JSR R4, INITM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$: JSR PC, PHYADR ;GET PHYSICAL ADDRESS INTO R0
COM RO ;COMPLEMENT THE ADR
2$: ADD #2, RO ;+2 TO DATA--ADR GOES DOWN SO COM GOES UP

```

L04

MAINDEC-11-DZQMC-B-D:
DZQMCB.P11 T3

0-124K MEMORY EXERCISER, 16K VER
WRITE 1'S COMPLEMENT VALUE OF ADDRESS INTO ADDRESS.

MACY11 27(732) 10-SEP-76 12:01 PAGE 51

```

2076 005720 010042          MOV    R0,    -(R2)    ;PUT DATA INTO MEMORY
2077 005722 030502          BIT    R5,    R2      ;CHECK FOR END OF A BLOCK.
2078 005724 001373          BNE   2$,    ;BRANCH IF MORE IN CURRENT BLOCK.
2079 005726 004767 010244    JSR   PC,    MMDOWN  ;FIND NEXT BLOCK AND LOOP TO 1$.
2080
2081                      ;* CHECK COMPLEMENT DATA WRITTEN DOWN
2082                      ;* UPWARDS WORD ADDRESSING.
2083 005732 004467 006772    JSR   R4,    INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2084 005736 004767 0104.4    3$:  JSR   PC,    PHYADR ;GET PHYSICAL ADDRESS INTO R0
2085 005742 005100          COM   R0      ;COMPLEMENT IT
2086 005744          4$:
2087 005744 012201          MOV   (R2)+,  R1     ;GET THE DATA FROM MEMORY UNDER TEST.
2088 005746 020001          CMP   R0,    R1     ;COMPARE THE CHECK WORD WITH THE DATA READ.
2089 005750 001405          BEQ  65$,    ;BRANCH OVER ERROR CALL IF GOOD DATA.
2090 005752 004767 012626    64$:  JSR   PC,    SPRT2   ;SET UP VALUES FOR ERROR PRINTING.
2091 005756 004767 014054    JSR   PC,    $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2092 005762 000002          .WORD 2          ;ERROR TYPE CODE.
2093
2094 005764 162700 000002    65$:  SUB   #2,    R0     ;COUNT DOWN WITH ADDRESS
2095 005770 030502          BIT   R5,    R2     ;CHECK FOR END OF A BLOCK.
2096 005772 001364          BNE  4$,    ;BRANCH IF MORE IN CURRENT BLOCK.
2097 005774 004767 007506    JSR   PC,    MMUP   ;FIND NEXT BLOCK AND LOOP TO 3$.
2098
2099                      ;*****
2100                      ;*TEST 4 WRITE BANK # INTO ALL ADDRESSES IN A 4K BANK
2101                      ;* R0 = DATA WRITTEN INTO MEMORY (SHOULD BE)
2102                      ;* R1 = DATA READ FROM MEMORY (WAS)
2103                      ;* R2 = VIRTUAL ADDRESS
2104                      ;* R3 = NOT USED
2105                      ;* R4 = NOT USED
2106                      ;* R5 = BLOCK BOUNDARY BIT MASK.
2107                      ;*****
2108 006000          TST4:
2109 006000 004567 013042    JSR   R5,    $SCOPE ;GO TO SCOPE ROUTINE.
2110 006004 000000          .WORD 0          ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
2111 006006 012767 000004 173172  MOV   #STN-1,$STEN ;;SET TEST NUMBER IN MAIL BOX
2112                      ;* UPWARDS BYTE ADDRESSING.
2113 006014 004467 00671C    JSR   R4,    INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2114 006020 004767 010406    1$:  JSR   PC,    BANKNO ;GET THE BANK NUMBER INTO R0
2115 006024 110022          2$:  MOVB  (R2)+,  R1     ;WRITE BANK # INTO ALL ADDRESSES
2116 006026 030502          BIT   R5,    R2     ;CHECK FOR END OF A BLOCK.
2117 006030 001375          BNE  2$,    ;BRANCH IF MORE IN CURRENT BLOCK.
2118 006032 004767 007450    JSR   PC,    MMUP   ;FIND NEXT BLOCK AND LOOP TO 1$.
2119
2120                      ;* CHECK THAT DATA WRITTEN ABOVE CAN BE READ
2121                      ;* UPWARDS BYTE ADDRESSING.
2122 006036 004467 006666    JSR   R4,    INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2123 006042 004767 010364    3$:  JSR   PC,    BANKNO ;GET THE BANK NUMBER INTO R0
2124 006046 112201          4$:  MOVB  (R2)+,  R1     ;READ THE DATA OUT OF MEMORY
2125 006050 020001          CMP   R0,    R1     ;COMPARE THE CHECK WORD WITH THE DATA READ.
2126 006052 001405          BEQ  65$,    ;BRANCH OVER ERROR CALL IF GOOD DATA.
2127 006054 004767 012506    64$:  JSR   PC,    SPRT1   ;SET UP VALUES FOR ERROR PRINTING.
2128 006060 004767 013752    JSR   PC,    $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2129 006064 000003          .WORD 3          ;ERROR TYPE CODE.
2130
2131 006066 030502          65$:  BIT   R5,    R2     ;CHECK FOR END OF A BLOCK.

```

M04

MAINDEC-11-DZQMC-B-D:
DZQMCB.P11 T4

0-124K MEMORY EXERCISER, 16K VER
WRITE BANK # INTO ALL ADDRESSES IN A 4K BANK

MACY11 27(732) 10-SEP-76 12:01 PAGE 52

```

2132 006070 001366          BNE 4$          ;BRANCH IF MORE IN CURRENT BLOCK.
2133 006072 004767 007410 JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 3$.
2134
2135 ;*****
2136 ;*TEST 5 WRITE 1'S COMPLEMENT OF BANK #.
2137 ;* R0 = DATA WRITTEN INTO MEMORY (SHOULD BE)
2138 ;* R1 = DATA READ FROM MEMORY (WAS)
2139 ;* R2 = VIRTUAL ADDRESS
2140 ;* R3 = NOT USED
2141 ;* R4 = NOT USED
2142 ;* R5 = BLOCK BOUNDARY BIT MASK.
2143 ;*****
2144 006076
2145 006076 004567 012744 JSR R5, $SCOPE ;GO TO SCOPE ROUTINE.
2146 006102 000000          .WORD 0 ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
2147 006104 012767 000005 173074 MOV #STN-1,$STEN ;;SET TEST NUMBER IN MAIL BOX
2148 ;* DOWNWARDS BYTE ADDRESSING.
2149 006112 004467 007250 JSR R4, INITDN ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2150 006116 004767 010310 1$: JSR PC, BANKNO ;GET THE BANK # INTO R0
2151 006122 005100          COM R0 ;SET 1'S COMPLEMENT OF BANK #
2152 006124 110042          2$: MOVB R0, -(R2) ;PUT 1'S COM OF BANK # INTO MEMORY
2153 006126 030502          BIT R5, R2 ;CHECK FOR END OF A BLOCK.
2154 006130 001375          BNE 2$ ;BRANCH IF MORE IN CURRENT BLOCK.
2155 006132 004767 010040 JSR PC, MMDOWN ;FIND NEXT BLOCK AND LOOP TO 1$.
2156
2157 ;* CHECK THAT DATA WRITTEN CAN BE READ.
2158 ;* DOWNWARDS BYTE ADDRESSING.
2159 006136 004467 007224 JSR R4, INITDN ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2160 006142 004767 010264 3$: JSR PC, BANKNO ;GET THE BANK # INTO R0
2161 006146 005100          COM R0 ;SET 1'S COMPLEMENT OF BANK #
2162 006150 114201          4$: MOVB -(R2), R1 ;READ DATA OUT OF MEMORY
2163 006152 020001          CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
2164 006154 001405          BEQ 65$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
2165 006156 004767 012376 64$: JSR PC, SPRNTO ;SET UP VALUES FOR ERROR PRINTING.
2166 006162 004767 013650 JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2167 006166 000003          .WORD 3 ;ERROR TYPE CODE.
2168
2169 006170 030502          65$: BIT R5, R2 ;CHECK FOR END OF A BLOCK.
2170 006172 001366          BNE 4$ ;BRANCH IF MORE IN CURRENT BLOCK.
2171 006174 004767 007776 JSR PC, MMDOWN ;FIND NEXT BLOCK AND LOOP TO $TAG1.

```

```

2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187 006200
2188 006200 004567 012642
2189
2190 006204 000000
2191 006206 012767 000006 172772
2192 006214 016700 173366
2193 006220 004467 006504
2194 006224 010022
2195 006226 030502
2196 006230 001375
2197 005232 004767 007250
2198
2199
2200
2201
2202 006236
2203 006236 004567 012604
2204 006242 000000
2205 006244 012767 000007 172734
2206 006252 016700 173330
2207 006256 004467 006446
2208 006262
2209 006262 012201
2210 006264 020001
2211 006266 001405
2212 006270 004767 012310
2213 006274 004767 013536
2214 006300 000004
2215 006302
2216 006302 030502
2217 006304 001366
2218 006306 004767 007174
2219
2220
2221
2222 006312 032777 000400 172616
2223 006320 001416
2224 006322 017746 172610
2225 006326 042716 177740
2226 006332 022726 000006
2227 006336 001007
  
```

```

.SBTTL SECTION 2:      WORST CASE NOISE TESTS
*****
;* THESE TESTS WRITE MEMORY WORST CASE NOISE TEST PATTERNS THROUGHOUT
;* MEMORY AND CHECK THAT THEY CAN BE WRITTEN AND READ.
*****
;*TEST 6      WRITE A CONSTANT INTO MEMORY.
;* THE CONSTANT IS USER SELECTABLE (DEFAULT = 0).
;*
;* R0 = DATA WRITTEN INTO MEMORY (SHOULD BE)
;* R1 = DATA READ FROM MEMORY (WAS)
;* R2 = VIRTUAL ADDRESS
;* R3 = NOT USED
;* R4 = NOT USED
;* R5 = BLOCK BOUNDARY BIT MASK.
*****
TST6:      JSR      R5,      $SCOPE      ;GO TO SCOPE ROUTINE.
           .WORD      0              ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
           MOV      #STN-1,$TESTN      ;SET TEST NUMBER IN MAIL BOX
TST6A:    MOV      .CONST, R0          ;GET USER CONSTANT
           JSR      R4,      INITMM     ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$:       MOV      R0,      (R2)+      ;WRITE CONSTANT INTO MEMORY.
           BIT      R5,      R2        ;CHECK FOR END OF A BLOCK.
           BNE     1$              ;BRANCH IF MORE IN CURRENT BLOCK.
           JSR      PC,      MMUP      ;FIND NEXT BLOCK AND LOOP TO 1$.
*****
;*TEST 7      READ MEMORY AND COMPARE TO CONSTANT.
;* IMPORTANT: THIS TEST SHOULD NOT BE RUN WITHOUT FIRST RUNNING TEST $TN.
*****
TST7:      JSR      R5,      $SCOPE      ;GO TO SCOPE ROUTINE.
           .WORD      0              ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
           MOV      #STN-1,$TESTN      ;SET TEST NUMBER IN MAIL BOX
           MOV      .CONST, R0          ;GET USER CONSTANT
           JSR      R4,      INITMM     ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$:       MOV      (R2)+, R1          ;GET THE DATA FROM MEMORY UNDER TEST.
           CMP     R0,      R1          ;COMPARE THE CHECK WORD WITH THE DATA READ.
           BEQ     65$              ;BRANCH OVER ERROR CALL IF GOOD DATA.
64$:      JSR      PC,      SPRT2      ;SET UP VALUES FOR ERROR PRINTING.
           JSR      PC,      $ERROR     ;*** ERROR *** (GO TYPE A MESSAGE)
           .WORD      4              ;ERROR TYPE CODE.
65$:      BIT      R5,      R2        ;CHECK FOR END OF A BLOCK.
           BNE     1$              ;BRANCH IF MORE IN CURRENT BLOCK.
           JSR      PC,      MMUP      ;FIND NEXT BLOCK AND LOOP TO 1$.
;* SPECIAL CHECK TO SEE IF TEST 6 IS SELECTED THRU THE SWR.
;* ALLOWS THE OPERATOR TO SWITCH BACK AND FORTH BETWEEN TESTS 6 AND 7
;* BY SIMPLY "TOGGLING" SW00 WHEN SW01, SW02, AND SW08 ARE SET.
           BIT      #SW08,  $SWR      ;CHECK THAT LOOP ON TEST BIT SET
           BEQ     TST10             ;BRANCH IF NOT LOOP ON TEST
           MOV     $SWR, -(SP)        ;GET SWITCH REGISTER DATA.
           BIC     #177740, (SP)     ;CLEAR NON-TEST-NUMBER SWITCHES.
           CMP     #6,      (SP)+    ;CHECK IF TEST 6 IN SWITCHES.
           BNE     TST10             ;BRANCH IF NOT TEST 6
  
```

B05

MAINDEC-11-D2JMC-B-C:
D2JMCB.P11

3-124K MEMORY EXERCISER, 16K VER
READ MEMORY AND COMPARE TO CONSTANT.

MACY11 27(732) 10-SEP-76 12:01 PAGE 54

006340 162767
006346 162767
006354 000000
006356 004567
006358 000000
006359 012767
006354 000010 172614
006372 016704
006376 004767
006402 012400
006404 001420
006406 004467 006316
006412 010012
006414 012201
006416 020001
006420 001405
006422 004767 012156
006426 004767 013404
006432 000004
006434 030502
006436 001365
006440 004767 007042
006444 000004

000001 172534
000036 172532
000010 172614
011452
006316
012156
013404
007042

SUB #1 \$STNM :RESET TEST NUM
SUB #TST7-TST6,SLPADR :RESET LOOP ADR
BR TST6A ;GO TO TEST 6

: *TEST 10 WORSE CASE NOISE (PARITY) WORD TESTING
: * CHECK MEMORY WITH A SERIES OF PATTERNS

TST10:
JSR R5, SSCOPE :GO TO SCOPE ROUTINE.
.WORD 0 :NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
MOV #STN-1,\$STNM :SET TEST NUMBER IN MAIL BOX.
MOV MFPAT, R4 :INITIALIZE PATTERN TABLE POINTER.
PC, CKPMER :CHECK FOR NON-TRAP PARITY MEMORY ERRORS.
(R4)+, R0 :GET THE DATA PATTERN.
BEQ TST11 :BR IF END OF TABLE.
JSR R4, INITMM :INITIALIZE THE MEMORY ADDRESS POINTERS.
MOV R0, (R2) :PUT DATA PATTERN INTO MEMORY.
MOV (R2)+, R1 :GET THE DATA FROM MEMORY UNDER TEST.
CMP R0, R1 :COMPARE THE CHECK WORD WITH THE DATA READ.
BEQ 655 :BRANCH OVER ERROR CALL IF GOOD DATA.
JSR PC, SPRT2 :SET UP VALUES FOR ERROR PRINTING.
JSR PC, SERPR :*** ERROR *** (GO TYPE A MESSAGE).
.WORD 4 :ERROR TYPE CODE.
BIT R5, R2 :CHECK FOR END OF A BLOCK.
BNE 655 :BRANCH IF MORE IN CURRENT BLOCK.
JSR PC, MMUP :FIND NEXT BLOCK AND LOOP TO 25.
BR 15 :BR BACK TO DO NEXT PATTERN

:GO TO SCOPE ROUTINE.
:NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
:SET TEST NUMBER IN MAIL BOX.
:INITIALIZE PATTERN TABLE POINTER.
:CHECK FOR NON-TRAP PARITY MEMORY ERRORS.
:GET THE DATA PATTERN.
:BR IF END OF TABLE.
:INITIALIZE THE MEMORY ADDRESS POINTERS.
:PUT DATA PATTERN INTO MEMORY.
:GET THE DATA FROM MEMORY UNDER TEST.
:COMPARE THE CHECK WORD WITH THE DATA READ.
:BRANCH OVER ERROR CALL IF GOOD DATA.
:SET UP VALUES FOR ERROR PRINTING.
:*** ERROR *** (GO TYPE A MESSAGE).
:ERROR TYPE CODE.
:CHECK FOR END OF A BLOCK.
:BRANCH IF MORE IN CURRENT BLOCK.
:FIND NEXT BLOCK AND LOOP TO 25.
:BR BACK TO DO NEXT PATTERN

22257
22258
22259
22260
22261
22262
22263
22264
22265
22266
22267
22268
22269
22270
22271
22272
22273
22274
22275
22276
22277
22278
22279
22280
22281
22282
22283
22284
22285
22286
22287
22288
22289
22290
22291
22292
22293
22294
22295
22296
22297
22298
22299
23000
23001
23002
23003

006446
006446 004567 012374
006452 000000
006454 012767 000011 172524
006462 012700 177777
006466 004767 010000
006472 004467 006232
006476 000241
006500 004767 010006
006504 016201 177776
006510 103402
006512 020001
006514 001405
006516 004767 012062
006522 004767 013310
006526 000005
006530
006530 030502
006532 001361
006534 004767 006746

172524

```
*****  
: *TEST 11 ROTATE A "0" BIT THROUGH A FIELD OF ONES.  
*****  
†ST11:  
JSR R5, $SCOPE :GO TO SCOPE ROUTINE.  
.WORD 0 :NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.  
MOV #STN-1, $TESTN :SET TEST NUMBER IN MAIL BOX  
MOV #-1, RC :SET CHECK WORD  
JSR PC, SETCON :PUT THE CONTENTS OF RC IN ALL MEMORY.  
JSR R4, INITMM :INITIALIZE THE MEMORY ADDRESS POINTERS.  
1$: CLC :CLEAR CARRY BIT IN PSW  
JSR PC, ROTATE  
MOV -2(R2), R1 :GET RESULT  
BCS 63$ :BRANCH IF 'C' BIT WAS SET  
CMP R0, R1 :COMPARE THE CHECK WORD WITH THE DATA READ.  
BEQ 64$ :BRANCH OVER ERROR CALL IF GOOD DATA.  
63$: JSR PC, SPRT2 :SET UP VALUES FOR ERROR PRINTING.  
JSR PC, $ERROR :*** ERROR *** (GO TYPE A MESSAGE)  
.WORD 5 :ERROR TYPE CODE.  
64$: BIT R5, R2 :CHECK FOR END OF A BLOCK.  
BNE 1$ :BRANCH IF MORE IN CURRENT BLOCK.  
JSR PC, MMUP :FIND NEXT BLOCK AND LOOP TO 1$.
```

```
*****  
: *TEST 12 ROTATE A "1" BIT THROUGH A FIELD OF ZEROS  
*****  
†ST12:  
JSR R5, $SCOPE :GO TO SCOPE ROUTINE.  
.WORD 0 :NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.  
MOV #STN-1, $TESTN :SET TEST NUMBER IN MAIL BOX  
CLR R0 :SET CHECK WORD  
JSR PC, SETCON :PUT THE CONTENTS OF RC IN ALL MEMORY  
JSR R4, INITMM :INITIALIZE THE MEMORY ADDRESS POINTERS.  
1$: SEC :SET 'C' BIT IN PSW  
JSR PC, ROTATE :GO ROTATE '1' BIT  
MOV -2(R2), R1 :GET RESULT  
BCC 63$ :BRANCH IF 'C' IS CLEAR  
CMP R0, R1 :COMPARE THE CHECK WORD WITH THE DATA READ.  
BEQ 64$ :BRANCH OVER ERROR CALL IF GOOD DATA.  
63$: JSR PC, SPRT2 :SET UP VALUES FOR ERROR PRINTING.  
JSR PC, $ERROR :*** ERROR *** (GO TYPE A MESSAGE)  
.WORD 5 :ERROR TYPE CODE.  
64$: BIT R5, R2 :CHECK FOR END OF A BLOCK.  
BNE 1$ :BRANCH IF MORE IN CURRENT BLOCK.  
JSR PC, MMUP :FIND NEXT BLOCK AND LOOP TO 1$.
```

172432

2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359

006630
005630 004567 012212
006634 000377

006636 000167 000442

006642 012767 000013 172336
006650 012700 177777
006654 005003
006656 004467 006046
006662 005100
006664 005103
006666 012704 000010
006672 010022
006674 010322
006676 010022
006700 010322

006702 010022
006704 010322
006706 010022
006710 010322

006712 010022
006714 010322
006716 010022
006720 010322

006722 010022
006724 010322
006726 010022
006730 010322

006732 005304
006734 001356
006736 030502
006740 001350
006742 004767 006540

006746 005000
006750 004467 005754
006754 012704 000040
006760
006760 012201
006762 020001
006764 001405
006766 004767 011612
006772 004767 013040
006776 000006
007000

```
*****
*TEST 13      1 XOR 8 TEST PATTERN
*****
TST13:
JSR    R5,    $SCOPE ;GO TO SCOPE ROUTINE.
.WORD  377       ;MINIMUM BLOCK SIZE OF 128. WORDS
                     ;REQUIRED FOR THIS TEST.
JMP    TST14     ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                     ;AVAILABLE FOR TEST.
MOV    #STN-1,$TESTN ;SET TEST NUMBER IN MAIL BOX
MOV    #-1,    RO  ;SET UP CHECK WORD.
CLR    R3       ;SET UP COM DATA REG
JSR    R4,    INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$:    COM     RO
      COM     R3
      MOV    #6, R4 ;SET 128. WORD COUNTER
2$:    MOV    RO, (R2)+ ;WRITE 128. WORDS
      MOV    R3, (R2)+
      MOV    RO, (R2)+
      MOV    R3, (R2)+

      MOV    RO, (R2)+
      MOV    R3, (R2)+
      MOV    RO, (R2)+
      MOV    R3, (R2)+

      MOV    RO, (R2)+
      MOV    R3, (R2)+
      MOV    RO, (R2)+
      MOV    R3, (R2)+

      MOV    RO, (R2)+
      MOV    R3, (R2)+
      MOV    RO, (R2)+
      MOV    R3, (R2)+

      DEC    R4 ;DECREMENT 128. WORD COUNTER
      BNE   2$
      BIT   R5, R2 ;CHECK FOR END OF A BLOCK.
      BNE   1$ ;BRANCH IF MORE IN CURRENT BLOCK.
      JSR   PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 1$.

*****
* CHECK 1 XOR 8 TEST PATTERN WRITTEN ABOVE.
*****
CLR    RO ;CLEAR TEST WORD
JSR    R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
11$:   MOV    #32, R4 ;SET 128. WORD COUNTER
12$:   MOV    (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
      CMP    RO, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
      BEQ   65$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
64$:   JSR    PC, SPRINT2 ;SET UP VALUES FOR ERROR PRINTING.
      JSR    PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
      .WORD 6 ;ERROR TYPE CODE.
65$:
```



```

2360 007000 005100      COM      R0
2361 007002 012201      MOV      (R2)+, R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2362 007004 020001      CMP      R0, R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2363 007006 001405      BEQ      67$          ;BRANCH OVER ERROR CALL IF GOOD DATA.
2364 007010 004767 011570      JSR      PC, SPRT2    ;SET UP VALUES FOR ERROR PRINTING.
2365 007014 004767 013015      JSR      PC, SEPROR   ;*** ERROR *** (GO TYPE A MESSAGE)
2366 007020 000006      .WORD    6           ;ERROR TYPE CODE.
2367 007022
2368 007022 005100      COM      R0
2369 007024 012201      MOV      (R2)+, R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2370 007026 020001      CMP      R0, R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2371 007030 001405      BEQ      69$          ;BRANCH OVER ERROR CALL IF GOOD DATA.
2372 007032 004767 011546      JSR      PC, SPRT2    ;SET UP VALUES FOR ERROR PRINTING.
2373 007036 004767 012774      JSR      PC, SEPROR   ;*** ERROR *** (GO TYPE A MESSAGE)
2374 007042 000006      .WORD    6           ;ERROR TYPE CODE.
2375 007044
2376 007044 005100      COM      R0
2377 007046 012201      MOV      (R2)+, R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2378 007050 020001      CMP      R0, R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2379 007052 001405      BEQ      71$          ;BRANCH OVER ERROR CALL IF GOOD DATA.
2380 007054 004767 011524      JSR      PC, SPRT2    ;SET UP VALUES FOR ERROR PRINTING.
2381 007060 004767 012752      JSR      PC, SEPROR   ;*** ERROR *** (GO TYPE A MESSAGE)
2382 007064 000006      .WORD    6           ;ERROR TYPE CODE.
2383 007066
2384 007066 005100      COM      R0           ;COMPLEMENT TEST DATA
2385 007070 005304      DEC      R4           ;DECREMENT 128. WORD COUNTER
2386 007072 001332      BNE      12$
2387 007074 005100      COM      R0           ;COMPLEMENT TEST DATA
2388 007076 030502      BIT      R5, R2      ;CHECK FOR END OF A BLOCK.
2389 007100 001325      BNE      11$          ;BRANCH IF MORE IN CURRENT BLOCK.
2390 007102 004767 00640C      JSR      PC, MMUP    ;FIND NEXT BLOCK AND LOOP TO 11$.
2391 007102
2392
2393
2394
2395 007106 004467 005616      JSR      R4, INITMM   ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2396 007112 012704 000040 21$: MOV      #32, R4      ;SET UP 128. WORD BLOCK COUNTER
2397 007116 005122 22$: COM      (R2)+      ;COMPLEMENT PATTERN
2398 007120 005122
2399 007122 005122
2400 007124 005122
2401 007126 005304      DEC      R4           ;CHECK FOR 128. WORDS DONE.
2402 007130 001372      BNE      22$          ;BRANCH IF MORE
2403 007132 030502      BIT      R5, R2      ;CHECK FOR END OF A BLOCK.
2404 007134 001366      BNE      21$          ;BRANCH IF MORE IN CURRENT BLOCK.
2405 007136 004767 006344      JSR      PC, MMUP    ;FIND NEXT BLOCK AND LOOP TO 21$.
2406
2407
2408
2409
2410 007142 012700 177777
2411 007146 004467 005556
2412 007152 012704 000040 31$: MOV      #32, R4      ;SET 128. WORD COUNTER
2413 007156
2414 007156 012201 32$: MOV      (R2)+, R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2415 007160 020001      CMP      R0, R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.

```

①

F05

MAINDEC-11-DZQMC-8-0:
DZQMC8.P11

0-124K MEMORY EXERCISER, 16K VER
1 XOR 8 TEST PATTERN

MACY11 27(732) 10-SEP-76 12:01 PAGE 58

2416	007162	001405								
2417	007164	004767	011414	72S:	BEQ	73S				: BRANCH OVER ERROR CALL IF GOOD DATA.
2418	007170	004767	012642		JSR	PC.	SPRNT2			: SET UP VALUES FOR ERROR PRINTING.
2419	007174	000006			JSR	PC.	\$ERROR			: *** ERROR *** (GO TYPE A MESSAGE)
2420	007176			73S:	.WORD	6				: ERROR TYPE CODE.
2421	007200	005100			COM	R0				
2422	007206	012201			MOV	(R2)+	R1			: GET THE DATA FROM MEMORY UNDER TEST.
2423	007208	020001			CMP	R0	R1			: COMPARE THE CHECK WORD WITH THE DATA READ.
2424	007204	001405			BEQ	75S				: BRANCH OVER ERROR CALL IF GOOD DATA.
2425	007208	004767	011372	74S:	JSR	PC.	SPRNT2			: SET UP VALUES FOR ERROR PRINTING.
2426	007212	004767	012620		JSR	PC.	\$ERROR			: *** ERROR *** (GO TYPE A MESSAGE)
2427	007216	000006			.WORD	6				: ERROR TYPE CODE.
2428	007220	005100		75S:	COM	R0				
2429	007222	012201			MOV	(R2)+	R1			: GET THE DATA FROM MEMORY UNDER TEST.
2430	007224	020001			CMP	R0	R1			: COMPARE THE CHECK WORD WITH THE DATA READ.
2431	007226	001405			BEQ	77S				: BRANCH OVER ERROR CALL IF GOOD DATA.
2432	007230	004767	011350	76S:	JSR	PC.	SPRNT2			: SET UP VALUES FOR ERROR PRINTING.
2433	007234	004767	012576		JSR	PC.	\$ERROR			: *** ERROR *** (GO TYPE A MESSAGE)
2434	007240	000006			.WORD	6				: ERROR TYPE CODE.
2435	007242			77S:	COM	R0				
2436	007244	005100			MOV	(R2)+	R1			: GET THE DATA FROM MEMORY UNDER TEST.
2437	007246	012201			CMP	R0	R1			: COMPARE THE CHECK WORD WITH THE DATA READ.
2438	007248	020001			BEQ	79S				: BRANCH OVER ERROR CALL IF GOOD DATA.
2439	007250	001405			JSR	PC.	SPRNT2			: SET UP VALUES FOR ERROR PRINTING.
2440	007252	004767	011326	78S:	JSR	PC.	\$ERROR			: *** ERROR *** (GO TYPE A MESSAGE)
2441	007254	004767	012554		JSR	PC.				: ERROR TYPE CODE.
2442	007256	000006			.WORD	6				
2443	007262			79S:	COM	R0				: COMPLEMENT TEST DATA
2444	007264	005100			DEC	R4				: DECREMENT 128. WORD COUNTER
2445	007266	005304			BNE	32S				
2446	007270	001332			COM	R0				: COMPLEMENT TEST DATA
2447	007272	005100			BIT	R5	R2			: CHECK FOR END OF A BLOCK.
2448	007274	030502			BNE	31S				: BRANCH IF MORE IN CURRENT BLOCK.
2449	007276	001325			JSR	PC.	MMUP			: FIND NEXT BLOCK AND LOOP TO 31S.
2450	007278	004767	006202							
2451	007300									

sep

2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507

007304
007304 004567 011536
007310 000777

007312 000167 000320

007316 012767 000014 171662
007324 005000
007326 012703 177777
007332 004467 005372
007336 004767 007216
007342 030502
007344 001374
007346 004767 006134

007352 005000
007354 004467 005350
007360 012704 000100
007364
007364 012201
007366 020001
007370 001405
007372 004767 011206
007376 004767 012434
007402 000007
007404
007404 012201
007406 020001
007410 001405
007412 004767 011136
007416 004767 012414
007422 000007
007424
007424 012201
007426 020001
007430 001405
007432 004767 011146
007436 004767 012374
007442 000007
007444
007444 012201
007446 020001
007450 001405
007452 004767 011126
007456 004767 012354
007462 000007
007464
007464 005100
007466 005304
007470 001335

```
*****
: *TEST 14      3 XOR 9 TEST PATTERN.
*****
†TST14:
      JSR      R5      $SCOPE ;GO TO SCOPE ROUTINE.
      .WORD   777
      JMP      TST15 ;MINIMUM BLOCK SIZE OF 256. WORDS
                        ;REQUIRED FOR THIS TEST.
                        ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                        ;AVAILABLE FOR TEST.
      MOV      #STN-1,$TESTN ;SET TEST NUMBER IN MAIL BOX
      CLR      R0 ;SET UP TEST DATA
      MOV      #-1, R3 ;SET COM DATA REG
      JSR      R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
      JSR      PC, W3X9 ;WRITE 256. WORD BLOCK WITH 3 XOR 9 PAT.
      BIT      R5, R2 ;CHECK FOR END OF A BLOCK.
      BNE     1$, ;BRANCH IF MORE IN CURRENT BLOCK.
      JSR      PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 1$.

*****
: * CHECK 3 XOR 9 TEST PATTERN WRITTEN ABOVE
*****
      CLR      R0 ;SET CHECK WORD
      JSR      R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
      MOV      #64, R4 ;SET 256. WORD COUNTER
11$:
12$:
      MOV      (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
      CMP      R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
      BEQ     65$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
      JSR      PC, SPRT2 ;SET UP VALUES FOR ERROR PRINTING.
      JSR      PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
      .WORD   7 ;ERROR TYPE CODE.
64$:
65$:
      MOV      (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
      CMP      R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
      BEQ     67$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
      JSR      PC, SPRT2 ;SET UP VALUES FOR ERROR PRINTING.
      JSR      PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
      .WORD   7 ;ERROR TYPE CODE.
66$:
67$:
      MOV      (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
      CMP      R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
      BEQ     69$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
      JSR      PC, SPRT2 ;SET UP VALUES FOR ERROR PRINTING.
      JSR      PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
      .WORD   7 ;ERROR TYPE CODE.
68$:
69$:
      MOV      (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
      CMP      R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
      BEQ     71$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
      JSR      PC, SPRT2 ;SET UP VALUES FOR ERROR PRINTING.
      JSR      PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
      .WORD   7 ;ERROR TYPE CODE.
70$:
71$:
      COM      R0 ;COMPLEMENT CHECK WORD
      DEC     R4 ;DECREMENT 256. WORD COUNTER
      BNE     12$
```

H05

MAINDEC-11-DZQMC-8-D:
DZQMCB.P11 T14

0-124K MEMORY EXERCISER, 16K VER
3 XOR 9 TEST PATTERN.

MACY11 27(732) 10-SEP-76 12:01 PAGE 60

```

2508 007472 005100          CUM   R0          ;COMPLEMENT CHECK WORD
2509 007474 030502          BIT   R5          ;CHECK FOR END OF A BLOCK.
2510 007476 001330          BNE  11$         ;BRANCH IF MORE IN CURRENT BLOCK.
2511 007500 004767 006002  JSR   PC,        MMUP ;FIND NEXT BLOCK AND LOOP TO 11$.
2512
2513
2514
2515
2516 007504 005000          CLR   R0          ;*****
2517 007506 004467 005216  JSR   R4,        INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2518 007512 012704 000100 21$: MOV  #64.,     R4      ;SET 256. WORD COUNTER
2519 007516 012703 000004 22$: MOV  #4,      R3      ;SET 4 WORD COUNTER
2520 007522
2521 007522 012201          MOV  (R2)+,     R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2522 007524 020001          CMP  R0,        R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2523 007526 001405          BEQ  73$         ;BRANCH OVER ERROR CALL IF GOOD DATA.
2524 007530 004767 011050 72$: JSR   PC,        SPRT2  ;SET UP VALUES FOR ERROR PRINTING.
2525 007534 004767 012276  JSR   PC,        $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2526 007540 000007          .WORD 7          ;ERROR TYPE CODE.
2527 007542
2528 007542 005100          COM  R0          ;COMPLEMENT CHECK WORD
2529 007544 005142          COM  -(R2)       ;COMPLEMENT TEST DATA
2530 007546 012201          MOV  (R2)+,     R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2531 007550 020001          CMP  R0,        R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2532 007552 001405          BEQ  75$         ;BRANCH OVER ERROR CALL IF GOOD DATA.
2533 007554 004767 011024 74$: JSR   PC,        SPRT2  ;SET UP VALUES FOR ERROR PRINTING.
2534 007560 004767 012252  JSR   PC,        $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2535 007564 000007          .WORD 7          ;ERROR TYPE CODE.
2536 007566
2537 007566 005100          COM  R0          ;COMPLEMENT CHECK WORD
2538 007570 005142          COM  -(R2)       ;COMPLEMENT TEST DATA
2539 007572 012201          MOV  (R2)+,     R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2540 007574 020001          CMP  R0,        R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2541 007576 001405          BEQ  77$         ;BRANCH OVER ERROR CALL IF GOOD DATA.
2542 007600 004767 011000 76$: JSR   PC,        SPRT2  ;SET UP VALUES FOR ERROR PRINTING.
2543 007604 004767 012226  JSR   PC,        $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2544 007610 000007          .WORD 7          ;ERROR TYPE CODE.
2545 007612
2546 007612 005303          DEC  R3          ;DECREMENT 4 WORD COUNTER
2547 007614 001342          BNE  23$         ;BR IF NOT DONE.
2548 007616 005100          COM  R0          ;COMPLEMENT CHECK WORD
2549 007620 005304          DEC  R4          ;DECREMENT 256. WORD COUNTER
2550 007622 001335          BNE  22$         ;BR IF NOT DONE.
2551 007624 005100          COM  R0          ;COMPLEMENT CHECK WORD
2552 007626 030502          BIT  R5          ;CHECK FOR END OF A BLOCK.
2553 007630 001330          BNE  21$         ;BRANCH IF MORE IN CURRENT BLOCK.
2554 007632 004767 005650  JSR   PC,        MMUP ;FIND NEXT BLOCK AND LOOP TO 21$.

```

```

25555
25556
25557
25558 007636
25559 007636 004567 011204
25560 007636 000777
25561 007642
25562 007644 000167 000324
25563
25564 00765C 012767 000C15 171330
25565 007656 012700 177777
25566 007652 005003
25567 007664 004467 005040
25568 007670 004767 006664
25569 007674 000502
25570 007676 001374
25571 00770C 004767 005602
25572
25573
25574
25575
25576
25577 007704 012700 177777
25578 007710 004467 005014
25579 007714 012704 000100
25580 007720
25581 007720 012201
25582 007722 020001
25583 007724 001405
25584 007726 004767 010652
25585 007732 004767 012100
25586 007736 000007
25587 007740
25588 007740 012201
25589 007742 020001
25590 007744 001405
25591 007746 004767 010622
25592 007752 004767 012060
25593 007756 000007
25594 007760
25595 007760 012201
25596 007762 020001
25597 007764 001405
25598 007766 004767 010612
25599 007772 004767 012040
2600 007776 000007
2601 010000
2602 010000 012201
2603 010002 020001
2604 010004 001405
2605 010006 004767 010572
2606 010012 004767 012020
2607 010016 000007
2608 010020
2609 010020 005100
2610 010022 005304

```

```

:*****
:*TEST 15      COMPLEMENT 3 XOR 9 TEST PATTERN
:*****
†TST15:
      JSR      R5,      $SCOPE ;GO TO SCOPE ROUTINE.
      .WORD    777
      JMP      TST16 ;MINIMUM BLOCK SIZE OF 256. WORDS
                        ;REQUIRED FOR THIS TEST.
                        ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                        ;AVAILABLE FOR TEST.
      MOV      #STN-1,$TESTN ;SET TEST NUMBER IN MAIL BOX
      MOV      #-1,   RO      ;SET UP TEST DATA
      CLR      R3      ;SET COM DATA REG
      JSR      R4,      INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$:    JSR      PC,      W3X9   ;WRITE 256. WORD BLOCK WITH 3 XOR 9 PAT.
      BIT      R5,      R2     ;CHECK FOR END OF A BLOCK.
      BNE     1$,      ;BRANCH IF MORE IN CURRENT BLOCK.
      JSR      PC,      MMUP   ;FIND NEXT BLOCK AND LOOP TO 1$.

:*****
:* CHECK COMPLEMENTED 3 XOR 9 TEST PATTERN WRITTEN ABOVE.
:*****
      MOV      #-1,   RO      ;SET CHECK WORD
      JSR      R4,      INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
11$:   MOV      #64.,  R4      ;SET 256. WORD COUNTER
12$:
      MOV      (R2)+,  R1     ;GET THE DATA FROM MEMORY UNDER TEST.
      CMP      RO,    R1     ;COMPARE THE CHECK WORD WITH THE DATA READ.
      BEQ     65$,    ;BRANCH OVER ERROR CALL IF GOOD DATA.
64$:   JSR      PC,      SPRT2  ;SET UP VALUES FOR ERROR PRINTING.
      JSR      PC,      $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
      .WORD    7 ;ERROR TYPE CODE.
65$:
      MOV      (R2)+,  R1     ;GET THE DATA FROM MEMORY UNDER TEST.
      CMP      RO,    R1     ;COMPARE THE CHECK WORD WITH THE DATA READ.
      BEQ     67$,    ;BRANCH OVER ERROR CALL IF GOOD DATA.
66$:   JSR      PC,      SPRT2  ;SET UP VALUES FOR ERROR PRINTING.
      JSR      PC,      $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
      .WORD    7 ;ERROR TYPE CODE.
67$:
      MOV      (R2)+,  R1     ;GET THE DATA FROM MEMORY UNDER TEST.
      CMP      RO,    R1     ;COMPARE THE CHECK WORD WITH THE DATA READ.
      BEQ     69$,    ;BRANCH OVER ERROR CALL IF GOOD DATA.
68$:   JSR      PC,      SPRT2  ;SET UP VALUES FOR ERROR PRINTING.
      JSR      PC,      $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
      .WORD    7 ;ERROR TYPE CODE.
69$:
      MOV      (R2)+,  R1     ;GET THE DATA FROM MEMORY UNDER TEST.
      CMP      RO,    R1     ;COMPARE THE CHECK WORD WITH THE DATA READ.
      BEQ     71$,    ;BRANCH OVER ERROR CALL IF GOOD DATA.
70$:   JSR      PC,      SPRT2  ;SET UP VALUES FOR ERROR PRINTING.
      JSR      PC,      $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
      .WORD    7 ;ERROR TYPE CODE.
71$:
      COM     RO
      DEC     R4 ;COMPLEMENT CHECK WORD
                        ;DECREMENT 256. WORD COUNTER

```

```

2611 010024 J01335          BNE      12$
2612 010026 005100          COM      R0
2613 010030 030502          BIT      R5      R2      ;COMPLEMENT CHECK WORD
2614 010032 001330          BNE      11$      ;CHECK FOR END OF A BLOCK.
2615 010034 004767 005446  JSR      PC,      MMUP   ;BRANCH IF MORE IN CURRENT BLOCK.
                               ;FIND NEXT BLOCK AND LOOP TO 11$.
2616
2617
2618
2619
2620
2621 010040 012700 :77777
2622 010044 004467 004660          MOV      #-1,      R0      ;SET UP CHECK WORD.
2623 010050 012704 000100          JSR      R4,      INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2624 010054 012703 000004          MOV      #64.,     R4      ;SET 256. WORD COUNTER
2625 010060          MOV      #4,      R3      ;SET 4 WORD COUNTER
2626 010062 012201          MOV      (R2)+,    R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2627 010064 001405          CMP      R0,      R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2628 010066 004767 010512          BEQ      73$      ;BRANCH OVER ERROR CALL IF GOOD DATA.
2629 010072 004767 011740          JSR      PC,      SPRT2   ;SET UP VALUES FOR ERROR PRINTING.
2630 010076 000007          JSR      PC,      $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2631 010100          .WORD      7          ;ERROR TYPE CODE.
2632
2633
2634
2635
2636
2637 010102 005100          COM      R0      ;COMPLEMENT CHECK WORD
2638 010104 005142          COM      -(R2)     ;COMPLEMENT TEST DATA
2639 010106 012201          MOV      (R2)+,    R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2640 010108 020001          CMP      R0,      R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2641 010110 001405          BEQ      75$      ;BRANCH OVER ERROR CALL IF GOOD DATA.
2642 010112 004767 010466          JSR      PC,      SPRT2   ;SET UP VALUES FOR ERROR PRINTING.
2643 010114 004767 011714          JSR      PC,      $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2644 010116 000007          .WORD      7          ;ERROR TYPE CODE.
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658

```

K05

MAINDEC-11-DZQMC-B-D:
DZQMCB.P11

T16

0-124K MEMORY EXERCISER, 16K VER
MODIFIED 3 XOR 9 PATTERN FOR PARITY MEMORY

MACY11 27(732) 10-SEP-76 12:01 PAGE 63

```

2659      ;*****
2660      ;*TEST 16      MODIFIED 3 XOR 9 PATTERN FOR PARITY MEMORY
2661      ;*****
2662      TST16:
2663      010174 004567 010646      JSR      R5,      $SCOPE      ;GO TO SCOPE ROUTINE.
2664      010200 000777      .WORD    777      ;MINIMUM BLOCK SIZE OF 256. WORDS
2665      ;REQUIRED FOR THIS TEST.
2666      010202 000167 000616      JMP      TST17      ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
2667      ;AVAILABLE FOR TEST.
2668      010206 012767 000016 170772      MOV      #STN-1,$TESTN      ;SET TEST NUMBER IN MAIL BOX
2669      010214 012700 000401      MOV      #401,  R0      ;SET UP PARITY "ALL ZEROS" PATTERN
2670      010220 012703 177777      MOV      #-1,   R3      ;SET COM DATA REG
2671      010224 004467 004500      JSR      R4,      INITMM      ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2672      010230 004767 006324      1$:     JSR      PC,      W3X9      ;WRITE 256. WORD BLOCK WITH 3 XOR 9 PAT.
2673      010234 030502      BIT      R5,      R2      ;CHECK FOR END OF A BLOCK.
2674      010236 001374      BNE     1$      ;BRANCH IF MORE IN CURRENT BLOCK.
2675      010240 004767 005242      JSR      PC,      MMUP      ;FIND NEXT BLOCK AND LOOP TO 1$.
2676
2677      ;*****
2678      ;* CHECK PARITY 3 XOR 9 PATTERN WRITTEN ABOVE.
2679      ;*****
2680      010244 012700 000401      MOV      #401,  R0      ;RESET PARITY "ALL ZEROS" PATTERN.
2681      010250 012703 177777      MOV      #-1,   R3      ;RESET PARITY ALL ONES PATTERN.
2682      010254 004467 004450      JSR      R4,      INITMM      ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2683      010260 012704 000100      11$:    MOV      #64.,  R4      ;SET 256. WORD COUNTER
2684      010264
2685      010264 012201      MOV      (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2686      010266 020001      CMP      R0,      R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2687      010270 001405      BEQ     65$      ;BRANCH OVER ERROR CALL IF GOOD DATA.
2688      010272 004767 010306      64$:    JSR      PC,      SPRT2      ;SET UP VALUES FOR ERROR PRINTING.
2689      010276 004767 011534      JSR      PC,      $ERROR      ;*** ERROR *** (GO TYPE A MESSAGE)
2690      010302 000007      .WORD    7      ;ERROR TYPE CODE.
2691
2692      010304 012201      MOV      (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2693      010306 020001      CMP      R0,      R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2694      010310 001405      BEQ     67$      ;BRANCH OVER ERROR CALL IF GOOD DATA.
2695      010312 004767 010266      66$:    JSR      PC,      SPRT2      ;SET UP VALUES FOR ERROR PRINTING.
2696      010316 004767 011514      JSR      PC,      $ERROR      ;*** ERROR *** (GO TYPE A MESSAGE)
2697      010322 000007      .WORD    7      ;ERROR TYPE CODE.
2698
2699      010324 012201      MOV      (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2700      010326 020001      CMP      R0,      R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2701      010330 001405      BEQ     69$      ;BRANCH OVER ERROR CALL IF GOOD DATA.
2702      010332 004767 010246      68$:    JSR      PC,      SPRT2      ;SET UP VALUES FOR ERROR PRINTING.
2703      010336 004767 011474      JSR      PC,      $ERROR      ;*** ERROR *** (GO TYPE A MESSAGE)
2704      010342 000007      .WORD    7      ;ERROR TYPE CODE.
2705
2706      010344 012201      MOV      (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2707      010346 020001      CMP      R0,      R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2708      010350 001405      BEQ     71$      ;BRANCH OVER ERROR CALL IF GOOD DATA.
2709      010352 004767 010226      70$:    JSR      PC,      SPRT2      ;SET UP VALUES FOR ERROR PRINTING.
2710      010356 004767 011454      JSR      PC,      $ERROR      ;*** ERROR *** (GO TYPE A MESSAGE)
2711      010362 000007      .WORD    7      ;ERROR TYPE CODE.
2712
2713      010364 010046      71$:    MOV      R0,      -(SP)      ;SAVE R0
2714      010366 010300      MOV      R3,      R0      ;PUT R3 INTO R0

```

L05

MAINDEC-11-DZQMC-B-D:
DZQMCB.P11 T16

0-124K MEMORY EXERCISER, 16K VER
MODIFIED 3 XOR 9 PATTERN FOR PARITY MEMORY

MACY11 27(732) 10-SEP-76 12:01 PAGE 64

2715	010370	012603		MOV	(SP)+, R3	; PUT SAVED R0 INTO R3
2716	010372	005304		DEC	R4	; COUNT 256. WORDS
2717	010374	001333		BNE	12\$; BRANCH IF MORE
2718	010376	010046		MOV	R0, -(SP)	; SAVE R0
2719	010400	010300		MOV	R3, R0	; PUT R3 INTO R0
2720	010402	012603		MOV	(SP)+, R3	; PUT SAVED R0 INTO R3
2721	010404	030502		BIT	R5, R2	; CHECK FOR END OF A BLOCK.
2722	010406	001324		BNE	11\$; BRANCH IF MORE IN CURRENT BLOCK.
2723	010410	004767	005072	JSR	PC, MMUP	; FIND NEXT BLOCK AND LOOP TO 11\$.
2724						
2725						
2726						
2727						
2728	010414	012700	000401			
2729	010420	012703	177777			
2730	010424	004467	004300			
2731	010430	012704	000100			
2732	010434			21\$:	MOV	#401, R0 ; SET UP PARITY "ALL ZEROS" PATTERN.
2733	010434	012201		22\$:	MOV	#-1, R3 ; SET UP ALL ONES PATTERN.
2734	010436	020001			JSR	R4, INITMM ; INITIALIZE THE MEMORY ADDRESS POINTERS.
2735	010440	001405			MOV	#64., R4 ; SET 256. WORD COUNTER
2736	010442	004767	010136		MOV	(R2)+, R1 ; GET THE DATA FROM MEMORY UNDER TEST.
2737	010446	004767	011364	72\$:	CMP	R0, R1 ; COMPARE THE CHECK WORD WITH THE DATA READ.
2738	010452	000007			BEQ	73\$; BRANCH OVER ERROR CALL IF GOOD DATA.
2739	010454				JSR	PC, SPRT2 ; SET UP VALUES FOR ERROR PRINTING.
2740	010454	005100			JSR	PC, \$ERROR ; *** ERROR *** (GO TYPE A MESSAGE)
2741	010456	005142			.WORD	7 ; ERROR TYPE CODE.
2742	010460	012201		73\$:	COM	R0 ; COMPLEMENT CHECK WORD
2743	010462	020001			COM	-(R2) ; COMPLEMENT TEST DATA
2744	010464	001405			MOV	(R2)+, R1 ; GET THE DATA FROM MEMORY UNDER TEST.
2745	010466	004767	010112	74\$:	CMP	R0, R1 ; COMPARE THE CHECK WORD WITH THE DATA READ.
2746	010472	004767	011340		BEQ	75\$; BRANCH OVER ERROR CALL IF GOOD DATA.
2747	010476	000007			JSR	PC, SPRT2 ; SET UP VALUES FOR ERROR PRINTING.
2748	010500				JSR	PC, \$ERROR ; *** ERROR *** (GO TYPE A MESSAGE)
2749	010500	005100			.WORD	7 ; ERROR TYPE CODE.
2750	010502	005142		75\$:	COM	R0 ; COMPLEMENT CHECK WORD
2751	010504	012201			COM	-(R2) ; RESTORE DATA
2752	010506	020001			MOV	(R2)+, R1 ; GET THE DATA FROM MEMORY UNDER TEST.
2753	010510	001405			CMP	R0, R1 ; COMPARE THE CHECK WORD WITH THE DATA READ.
2754	010512	004767	010066	76\$:	BEQ	77\$; BRANCH OVER ERROR CALL IF GOOD DATA.
2755	010516	004767	011314		JSR	PC, SPRT2 ; SET UP VALUES FOR ERROR PRINTING.
2756	010522	000007			JSR	PC, \$ERROR ; *** ERROR *** (GO TYPE A MESSAGE)
2757	010524				.WORD	7 ; ERROR TYPE CODE.
2758	010524	012201		77\$:	MOV	(R2)+, R1 ; GET THE DATA FROM MEMORY UNDER TEST.
2759	010526	020001			CMP	R0, R1 ; COMPARE THE CHECK WORD WITH THE DATA READ.
2760	010530	001405			BEQ	79\$; BRANCH OVER ERROR CALL IF GOOD DATA.
2761	010532	004767	010046	78\$:	JSR	PC, SPRT2 ; SET UP VALUES FOR ERROR PRINTING.
2762	010536	004767	011274		JSR	PC, \$ERROR ; *** ERROR *** (GO TYPE A MESSAGE)
2763	010542	000007			.WORD	7 ; ERROR TYPE CODE.
2764	010544			79\$:	COM	R0 ; COMPLEMENT CHECK WORD
2765	010544	005100			COM	-(R2) ; COMPLEMENT TEST DATA
2766	010546	005142			MOV	(R2)+, R1 ; GET THE DATA FROM MEMORY UNDER TEST.
2767	010550	012201			CMP	R0, R1 ; COMPARE THE CHECK WORD WITH THE DATA READ.
2768	010552	020001			BEQ	81\$; BRANCH OVER ERROR CALL IF GOOD DATA.
2769	010554	001405			JSR	PC, SPRT2 ; SET UP VALUES FOR ERROR PRINTING.
2770	010556	004767	010022	80\$:	JSR	PC, SPRT2 ; SET UP VALUES FOR ERROR PRINTING.

M05

MAINDEC-11-DZJMC-8-C:
DZJMCB.P11

0-124K MEMORY EXERCISER, 16K VER
MODIFIED 3 XOR 9 PATTERN FOR PARITY MEMORY

MACY11 27(732) 10-SEP-76 12:01 PAGE 65

2771	010562	004767	011250	JSR	PC,		\$ERROR	*** ERROR *** (GO TYPE A MESSAGE)
2772	010566	000007		.WORD	7			;ERROR TYPE CODE.
2773	010570					81\$:		
2774	010570	005100		COM	RO			;COMPLEMENT CHECK WORD
2775	010572	005142		COM	-(R2)			;RESTORE DATA
2776	010574	012201		MOV	(R2)+,	R1		;GET THE DATA FROM MEMORY UNDER TEST.
2777	010576	020001		CMP	RO,	R1		;COMPARE THE CHECK WORD WITH THE DATA READ.
2778	010600	001405		BEQ	83\$;BRANCH OVER ERROR CALL IF GOOD DATA.
2779	010602	004767	007776	JSR	PC,		SPRNT2	;SET UP VALUES FOR ERROR PRINTING.
2780	010606	004767	011224	JSR	PC,		\$ERROR	*** ERROR *** (GO TYPE A MESSAGE)
2781	010612	000007		.WORD	7			;ERROR TYPE CODE.
2782	010614					83\$:		
2783	010614	012201		MOV	(R2)+,	R1		;GET THE DATA FROM MEMORY UNDER TEST.
2784	010616	020001		CMP	RO,	R1		;COMPARE THE CHECK WORD WITH THE DATA READ.
2785	010620	001405		BEQ	85\$;BRANCH OVER ERROR CALL IF GOOD DATA.
2786	010622	004767	007756	JSR	PC,		SPRNT2	;SET UP VALUES FOR ERROR PRINTING.
2787	010626	004767	011204	JSR	PC,		\$ERROR	*** ERROR *** (GO TYPE A MESSAGE)
2788	010632	000007		.WORD	7			;ERROR TYPE CODE.
2789	010634					85\$:		
2790	010634	005100		COM	RO			;COMPLEMENT CHECK WORD
2791	010636	005142		COM	-(R2)			;COMPLEMENT TEST DATA
2792	010640	012201		MOV	(R2)+,	R1		;GET THE DATA FROM MEMORY UNDER TEST.
2793	010642	020001		CMP	RO,	R1		;COMPARE THE CHECK WORD WITH THE DATA READ.
2794	010644	001405		BEQ	87\$;BRANCH OVER ERROR CALL IF GOOD DATA.
2795	010646	004767	007732	JSR	PC,		SPRNT2	;SET UP VALUES FOR ERROR PRINTING.
2796	010652	004767	011160	JSR	PC,		\$ERROR	*** ERROR *** (GO TYPE A MESSAGE)
2797	010656	000007		.WORD	7			;ERROR TYPE CODE.
2798	010660					87\$:		
2799	010660	005100		COM	RO			;COMPLEMENT CHECK WORD
2800	010662	005142		COM	-(R2)			;RESTORE DATA
2801	010664	012201		MOV	(R2)+,	R1		;GET THE DATA FROM MEMORY UNDER TEST.
2802	010666	020001		CMP	RO,	R1		;COMPARE THE CHECK WORD WITH THE DATA READ.
2803	010670	001405		BEQ	89\$;BRANCH OVER ERROR CALL IF GOOD DATA.
2804	010672	004767	007706	JSR	PC,		SPRNT2	;SET UP VALUES FOR ERROR PRINTING.
2805	010676	004767	011134	JSR	PC,		\$ERROR	*** ERROR *** (GO TYPE A MESSAGE)
2806	010702	000007		.WORD	7			;ERROR TYPE CODE.
2807	010704					89\$:		
2808	010704	012201		MOV	(R2)+,	R1		;GET THE DATA FROM MEMORY UNDER TEST.
2809	010706	020001		CMP	RO,	R1		;COMPARE THE CHECK WORD WITH THE DATA READ.
2810	010710	001405		BEQ	91\$;BRANCH OVER ERROR CALL IF GOOD DATA.
2811	010712	004767	007666	JSR	PC,		SPRNT2	;SET UP VALUES FOR ERROR PRINTING.
2812	010716	004767	011114	JSR	PC,		\$ERROR	*** ERROR *** (GO TYPE A MESSAGE)
2813	010722	000007		.WORD	7			;ERROR TYPE CODE.
2814	010724					91\$:		
2815	010724	005100		COM	RO			;COMPLEMENT CHECK WORD
2816	010726	005142		COM	-(R2)			;COMPLEMENT TEST DATA
2817	010730	012201		MOV	(R2)+,	R1		;GET THE DATA FROM MEMORY UNDER TEST.
2818	010732	020001		CMP	RO,	R1		;COMPARE THE CHECK WORD WITH THE DATA READ.
2819	010734	001405		BEQ	93\$;BRANCH OVER ERROR CALL IF GOOD DATA.
2820	010736	004767	007642	JSR	PC,		SPRNT2	;SET UP VALUES FOR ERROR PRINTING.
2821	010742	004767	011070	JSR	PC,		\$ERROR	*** ERROR *** (GO TYPE A MESSAGE)
2822	010746	000007		.WORD	7			;ERROR TYPE CODE.
2823	010750					93\$:		
2824	010750	005100		COM	RO			;COMPLEMENT CHECK WORD
2825	010752	005142		COM	-(R2)			;RESTORE DATA
2826	010754	012201		MOV	(R2)+,	R1		;GET THE DATA FROM MEMORY UNDER TEST.

N05

MAINDEC-11-DZQMC-B-D:
DZQMCB.P11

0-124K MEMORY EXERCISER, 16K VER
MODIFIED 3 XOR 9 PATTERN FOR PARITY MEMORY

MACY11 27(732) 10-SEP-76 12:01 PAGE 66

```

2827 010756 020001      CMP      R0,      R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2828 010760 001405      BEQ      95$,      ;BRANCH OVER ERROR CALL IF GOOD DATA.
2829 010762 004767 007616 94$:      JSR      PC,      SPRINT2 ;SET UP VALUES FOR ERPROR PRINTING.
2830 010766 004767 011044      JSR      PC,      $ERROR   ;*** ERROR *** (GO TYPE A MESSAGE)
2831 010772 000007      .WORD   7              ;ERROR TYPE CODE.
2832 010774          95$:
2833 010774 010046      MOV      R0,      -(SP)   ;SAVE R0
2834 010776 010300      MOV      R3,      R0      ;PUT R3 INTO R0
2835 011000 012603      MOV      (SP)+,   R3      ;PUT SAVED R0 INTO R3
2836 011002 005304      DEC      R4              ;DECREMENT 256. WORD COUNTER
2837 011004 001213      BNE     22$,      ;BRANCH IF MORE.
2838 011006 010946      MOV      R0,      -(SP)   ;SAVE R0
2839 011010 010300      MOV      R3,      R0      ;PUT R3 INTO R0
2840 011012 012603      MOV      (SP)+,   R3      ;PUT SAVED R0 INTO R3
2841 011014 030502      BIT      R5,      R2      ;CHECK FOR END OF A BLOCK.
2842 011016 001204      BNE     21$,      ;BRANCH IF MORE IN CURRENT BLOCK.
2843 011020 004767 004462      JSR      PC,      MMUP    ;FIND NEXT BLOCK AND LOOP TO 21$.
2844
2845 ;*****
2846 ;*TEST 17      COMPLEMENT PARITY 3 XOR 9 TEST PATTERN.
2847 ;*****
2848 011024          TST17:
2849 011024 004567 010016      JSR      R5,      $SCOPE  ;GO TO SCOPE ROUTINE.
2850 011030 000777      .WORD   777           ;MINIMUM BLOCK SIZE OF 256. WORDS
2851          ;REQUIRED FOR THIS TEST.
2852 011032 000167 000616      JMP      TST20        ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
2853          ;AVAILABLE FOR TEST.
2854 011036 012767 000017 170142      MOV      $STN-1,$TESTN ;SET TEST NUMBER IN MAIL BOX
2855 011044 012700 177777      MOV      #-1,      R0     ;SET UP ALL ONES PATTERN
2856 011050 012703 000401      MOV      #401,    R3     ;SET UP PARITY "ALL ZEROS" PATTERN
2857 011054 004467 003650      JSR      R4,      INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2858 011060 004767 005474 1$:      JSR      PC,      W3X9   ;WRITE 256. WORD BLOCK WITH 3 XOR 9 PAT.
2859 011064 030502      BIT      R5,      R2     ;CHECK FOR END OF A BLOCK.
2860 011066 001374      BNE     1$,      ;BRANCH IF MORE IN CURRENT BLOCK.
2861 011070 004767 004412      JSR      PC,      MMUP   ;FIND NEXT BLOCK AND LOOP TO 1$.
2862
2863 ;*****
2864 ;* CHECK COMPLEMENT PARITY 3 XOR 9 PATTERN WRITTEN ABOVE.
2865 ;*****
2866 011074 012700 177777      MOV      #-1,      R0     ;SET UP ALL ONES PATTERN
2867 011100 012703 000401      MOV      #401,    R3     ;SET UP PARITY "ALL ZEROS" PATTERN
2868 011104 004467 003620      JSR      R4,      INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2869 011110 012704 000100 11$:      MOV      #64.,    R4     ;SET 256. WORD COUNTER
2870 011114          12$:
2871 011114 012201      MOV      (R2)+,   R1     ;GET THE DATA FROM MEMORY UNDER TEST.
2872 011116 020001      CMP      R0,      R1     ;COMPARE THE CHECK WORD WITH THE DATA READ.
2873 011120 001405      BEQ     65$,      ;BRANCH OVER ERROR CALL IF GOOD DATA.
2874 011122 004767 007456 64$:      JSR      PC,      SPRINT2 ;SET UP VALUES FOR ERROR PRINTING.
2875 011126 004767 010704      JSR      PC,      $ERROR   ;*** ERROR *** (GO TYPE A MESSAGE)
2876 011132 000007      .WORD   7              ;ERROR TYPE CODE.
2877 011134          65$:
2878 011134 012201      MOV      (R2)+,   R1     ;GET THE DATA FROM MEMORY UNDER TEST.
2879 011136 020001      CMP      R0,      R1     ;COMPARE THE CHECK WORD WITH THE DATA READ.
2880 011140 001405      BEQ     66$,      ;BRANCH OVER ERROR CALL IF GOOD DATA.
2881 011142 004767 007436 66$:      JSR      PC,      SPRINT2 ;SET UP VALUES FOR ERROR PRINTING.
2882 011146 004767 010664      JSR      PC,      $ERROR   ;*** ERROR *** (GO TYPE A MESSAGE)

```

0-124 MEMORY EXERCISER, 16K VER
COMPLEMENT PARITY 3 XOR 9 TEST PATTERN.

0-124 MEMORY EXERCISER, 16K VER
COMPLEMENT PARITY 3 XOR 9 TEST PATTERN.

```

000007 011152 000007 .WORD 7 ;ERROR TYPE CODE.
575: MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
BEQ 69$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
585: JSR PC, SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
.WORD 7 ;ERROR TYPE CODE.
695: MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
BEQ 71$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
705: JSR PC, SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
.WORD 7 ;ERROR TYPE CODE.
715: MOV R0, -.SP) ;SAVE R0
MOV R3, R0 ;PUT R3 INTO R0
MOV (SP)+, R3 ;PUT SAVED R0 INTO R3
DEC R4 ;COUNT 256. WORDS
BNE 12$ ;BRANCH IF MORE
MOV R0, -.SP) ;SAVE R0
MOV R3, R0 ;PUT R3 INTO R0
MOV (SP)+, R3 ;PUT SAVED R0 INTO R3
BIT R5, R2 ;CHECK FOR END OF A BLOCK.
BNE 11$ ;BRANCH IF MORE IN CURRENT BLOCK.
JSR PC, MMLP ;FIND NEXT BLOCK AND LOOP TO 11$.

*****
* CHECK, COM, CHECK, COM, CHECK COMPLEMENTED PARITY 3 XOR 9 PATTERN.
*****
01244 012700 177777 MOV #-1, R0 ;SET UP ALL ONES PATTERN
01250 012703 000401 MOV #401, R3 ;SET UP PARITY "ALL ZEROS" PATTERN
01254 004467 003450 JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
01260 012704 000100 215: MOV #64, R4 ;SET 256. WORD COUNTER
225: MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
BEQ 73$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
725: JSR PC, SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
.WORD 7 ;ERROR TYPE CODE.
735: COM R0 ;COMPLEMENT CHECK WORD
COM -(R2) ;COMPLEMENT TEST DATA
MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
BEQ 75$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
745: JSR PC, SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
.WORD 7 ;ERROR TYPE CODE.
755: COM R0 ;COMPLEMENT CHECK WORD
COM -(R2) ;RESTORE DATA
MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.

```

C06

MAINDEC-11-DZQMC-B-D:
DZQMCB.P11

0-124K MEMORY EXERCISER, 16K VER
COMPLEMENT PARITY 3 XOR 9 TEST PATTERN.

MACY11 27(732) 10-SEP-76 12:01. PAGE 68

2933	011340	001405			BEQ	77\$:BRANCH OVER ERROR CALL IF GOOD DATA.
2934	011342	004767	007236	76\$:	JSR	PC,	SPRNT2			:SET UP VALUES FOR ERROR PRINTING.
2935	011346	004767	010464		JSR	PC,	\$ERROR			:*** ERROR *** (GO TYPE A MESSAGE)
2936	011352	000007			.WORD	7				:ERROR TYPE CODE.
2937	011354			77\$:						
2938	011354	012201			MOV	(R2)+,	R1			:GET THE DATA FROM MEMORY UNDER TEST.
2939	011356	020001			CMP	RO,	R1			:COMPARE THE CHECK WORD WITH THE DATA READ.
2940	011360	001405			BEQ	79\$:BRANCH OVER ERROR CALL IF GOOD DATA.
2941	011362	004767	007216	78\$:	JSR	PC,	SPRNT2			:SET UP VALUES FOR ERROR PRINTING.
2942	011366	004767	010444		JSR	PC,	\$ERROR			:*** ERROR *** (GO TYPE A MESSAGE)
2943	011372	000007			.WORD	7				:ERROR TYPE CODE.
2944	011374			79\$:						
2945	011374	005100			COM	RO				:COMPLEMENT CHECK WORD
2946	011376	005142			COM	-(R2)				:COMPLEMENT TEST DATA
2947	011400	012201			MOV	(R2)+,	R1			:GET THE DATA FROM MEMORY UNDER TEST.
2948	011402	020001			CMP	RO,	R1			:COMPARE THE CHECK WORD WITH THE DATA READ.
2949	011404	001405			BEQ	81\$:BRANCH OVER ERROR CALL IF GOOD DATA.
2950	011406	004767	007172	80\$:	JSR	PC,	SPRNT2			:SET UP VALUES FOR ERROR PRINTING.
2951	011412	004767	010420		JSR	PC,	\$ERROR			:*** ERROR *** (GO TYPE A MESSAGE)
2952	011416	000007			.WORD	7				:ERROR TYPE CODE.
2953	011420			81\$:						
2954	011420	005100			COM	RO				:COMPLEMENT CHECK WORD
2955	011422	005142			COM	-(R2)				:RESTORE DATA
2956	011424	012201			MOV	(R2)+,	R1			:GET THE DATA FROM MEMORY UNDER TEST.
2957	011426	020001			CMP	RO,	R1			:COMPARE THE CHECK WORD WITH THE DATA READ.
2958	011430	001405			BEQ	83\$:BRANCH OVER ERROR CALL IF GOOD DATA.
2959	011432	004767	007146	82\$:	JSR	PC,	SPRNT2			:SET UP VALUES FOR ERROR PRINTING.
2960	011436	004767	010374		JSR	PC,	\$ERROR			:*** ERROR *** (GO TYPE A MESSAGE)
2961	011442	000007			.WORD	7				:ERROR TYPE CODE.
2962	011444			83\$:						
2963	011444	012201			MOV	(R2)+,	R1			:GET THE DATA FROM MEMORY UNDER TEST.
2964	011446	020001			CMP	RO,	R1			:COMPARE THE CHECK WORD WITH THE DATA READ.
2965	011450	001405			BEQ	85\$:BRANCH OVER ERROR CALL IF GOOD DATA.
2966	011452	004767	007126	84\$:	JSR	PC,	SPRNT2			:SET UP VALUES FOR ERROR PRINTING.
2967	011456	004767	010354		JSR	PC,	\$ERROR			:*** ERROR *** (GO TYPE A MESSAGE)
2968	011462	000007			.WORD	7				:ERROR TYPE CODE.
2969	011464			85\$:						
2970	011464	005100			COM	RO				:COMPLEMENT CHECK WORD
2971	011466	005142			COM	-(R2)				:COMPLEMENT TEST DATA
2972	011470	012201			MOV	(R2)+,	R1			:GET THE DATA FROM MEMORY UNDER TEST.
2973	011472	020001			CMP	RO,	R1			:COMPARE THE CHECK WORD WITH THE DATA READ.
2974	011474	001405			BEQ	87\$:BRANCH OVER ERROR CALL IF GOOD DATA.
2975	011476	004767	007102	86\$:	JSR	PC,	SPRNT2			:SET UP VALUES FOR ERROR PRINTING.
2976	011502	004767	010330		JSR	PC,	\$ERROR			:*** ERROR *** (GO TYPE A MESSAGE)
2977	011506	000007			.WORD	7				:ERROR TYPE CODE.
2978	011510			87\$:						
2979	011510	005100			COM	RO				:COMPLEMENT CHECK WORD
2980	011512	005142			COM	-(R2)				:RESTORE DATA
2981	011514	012201			MOV	(R2)+,	R1			:GET THE DATA FROM MEMORY UNDER TEST.
2982	011516	020001			CMP	RO,	R1			:COMPARE THE CHECK WORD WITH THE DATA READ.
2983	011520	001405			BEQ	89\$:BRANCH OVER ERROR CALL IF GOOD DATA.
2984	011522	004767	007056	88\$:	JSR	PC,	SPRNT2			:SET UP VALUES FOR ERROR PRINTING.
2985	011526	004767	010304		JSR	PC,	\$ERROR			:*** ERROR *** (GO TYPE A MESSAGE)
2986	011532	000007			.WORD	7				:ERROR TYPE CODE.
2987	011534			89\$:						
2988	011534	012201			MOV	(R2)+,	R1			:GET THE DATA FROM MEMORY UNDER TEST.

2995	011536	020001		CMP	RO	R1	:COMPARE THE CHECK WORD WITH THE DATA READ.
2996	011540	001405		BEQ	91\$:BRANCH OVER ERROR CALL IF GOOD DATA.
2997	011542	004767	007036	JSR	PC.	SPRNT2	:SET UP VALUES FOR ERROR PRINTING.
2998	011546	004767	012264	JSR	PC.	\$ERROR	:*** ERROR *** (GO TYPE A MESSAGE)
2999	011552	000007		.WORD			:ERROR TYPE CODE.
3000	011554		91\$:				
3001	011554	005100		COM	RO		:COMPLEMENT CHECK WORD
3002	011556	005142		COM	-(R2)		:COMPLEMENT TEST DATA
3003	011560	012201		MOV	(R2)+,	R1	:GET THE DATA FROM MEMORY UNDER TEST.
3004	011562	020001		CMP	RO	R1	:COMPARE THE CHECK WORD WITH THE DATA READ.
3005	011564	001405		BEQ	93\$:BRANCH OVER ERROR CALL IF GOOD DATA.
3006	011566	004767	007012	JSR	PC.	SPRNT2	:SET UP VALUES FOR ERROR PRINTING.
3007	011572	004767	010240	JSR	PC.	\$ERROR	:*** ERROR *** (GO TYPE A MESSAGE)
3008	011576	000007		.WORD	7		:ERROR TYPE CODE.
3009	011600		93\$:				
3010	011600	005100		COM	RO		:COMPLEMENT CHECK WORD
3011	011602	005142		COM	-(R2)		:RESTORE DATA
3012	011604	012201		MOV	(R2)+,	R1	:GET THE DATA FROM MEMORY UNDER TEST.
3013	011606	020001		CMP	RO	R1	:COMPARE THE CHECK WORD WITH THE DATA READ.
3014	011610	001405		BEQ	95\$:BRANCH OVER ERROR CALL IF GOOD DATA.
3015	011612	004767	006756	JSR	PC.	SPRNT2	:SET UP VALUES FOR ERROR PRINTING.
3016	011616	004767	010214	JSR	PC.	\$ERROR	:*** ERROR *** (GO TYPE A MESSAGE)
3017	011622	000007		.WORD	7		:ERROR TYPE CODE.
3018	011624		95\$:				
3019	011624	010046		MOV	RO,	-(SP)	:SAVE RO
3020	011626	010300		MOV	R3,	RO	:PUT R3 INTO RO
3021	011630	012603		MOV	(SP)+,	R3	:PUT SAVED RO INTO R3
3022	011632	005304		DEC	R4		:DECREMENT 256. WORD COUNTER
3023	011634	001213		BNE	22\$:BRANCH IF MORE.
3024	011636	010046		MOV	RO,	-(SP)	:SAVE RO
3025	011640	010300		MOV	R3,	RO	:PUT R3 INTO RO
3026	011642	012603		MOV	(SP)+,	R3	:PUT SAVED RO INTO R3
3027	011644	030502		BIT	R5,	R2	:CHECK FOR END OF A BLOCK.
3028	011646	001204		BNE	21\$:BRANCH IF MORE IN CURRENT BLOCK.
3029	011650	004767	003632	JSR	PC.	MMUP	:FIND NEXT BLOCK AND LOOP TO 21\$.

E06

MAINDEC-11-DZQMC-B-D:
DZQMCB.P11 T20

0-124K MEMORY EXERCISER, 16k VER
8 XOR 13 TEST PATTERN

MACY11 27(732) 10-SEP-76 12:01 PAGE 70

```

3030
3031
3032
3033 011654
3034 011654 004567 007166
3035 011660 017777
3036
3037 011662 000167 000326
3038
3039 011666 012767 000020 167312
3040 011674 005000 .8X13:
3041 011676 004467 003026
3042 011702 012704 000040 1$:
3043 011706 005100
3044 011710 012703 000200 2$:
3045 011714 005100
3046 011716 010022 3$:
3047 011720 005303
3048 011722 001375
3049 011724 005304
3050 011726 001370
3051 011730 030502
3052 011732 001353
3053 011734 004767 003546
3054
3055
3056
3057
3058 011740 005000
3059 011742 004467 002762
3060 011746 012704 000040
3061 011752 005100
3062 011754 012703 000100
3063 011760 005100
3064 011762
3065 011762 012201
3066 011764 020001
3067 011766 001405
3068 011770 004767 006610
3069 011774 004767 010036
3070 012000 000010
3071 012002
3072 012002 012201
3073 012004 020001
3074 012006 001405
3075 012010 004767 006570
3076 012014 004767 010016
3077 012020 000010
3078 012022
3079 012022 005303
3080 012024 001356
3081 012026 005304
3082 012030 001351
3083 012032 030502
3084 012034 001344
3085 012036 004767 003444

```

```

:*****
;*TEST 20      8 XOR 13 TEST PATTERN
:*****
†T20:
      JSR      R5      $SCOPE ;GO TO SCOPE ROUTINE.
      .WORD    17777 ;MINIMUM BLOCK SIZE OF 4096. WORDS
                        REQUIRED FOR THIS TEST.
      JMP      TST21 ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                        AVAILABLE FOR TEST.
                        ;;SET TEST NUMBER IN MAIL BOX
      MOV      #STN-1,$TESTN
      CLR      R0
      JSR      R4      INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
      MOV      #32., R4 ;SET UP 4K LOOP COUNTER
      COM      R0
      MOV      #128., R3 ;EACH SMALL LOOP WRITES 128 WORDS
      COM      R0
      MOV      R0, (R2)+ ;WRITE INTO MEMORY ADDRESSES
      DEC      R3 ;DECREMENT WORD COUNT
      BNE     3$
      DEC      R4 ;DECREMENT 128. WORD COUNT
      BNE     2$
      BIT      R5, R2 ;CHECK FOR END OF A BLOCK.
      BNE     1$ ;BRANCH IF MORE IN CURRENT BLOCK.
      JSR      PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 1$.

:*****
;* CHECK 8 XOR 13 TEST PATTERN WRITTEN ABOVE
:*****
      CLR      R0 ;SET UP CHECK WORD.
      JSR      R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
      MOV      #32., R4 ;SET 4K WORD COUNTER
      COM      R0 ;COMPLEMENT TEST WORD
      MOV      #64., R3 ;SET 128 WORD COUNTER
      COM      R0 ;COMPLEMENT TEST WORD
      MOV      (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
      CMP      R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
      BEQ     65$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
      JSR      PC, SPRT2 ;SET UP VALUES FOR ERROR PRINTING.
      JSR      PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
      .WORD    10 ;ERROR TYPE CODE.
      MOV      (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
      CMP      R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
      BEQ     67$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
      JSR      PC, SPRT2 ;SET UP VALUES FOR ERROR PRINTING.
      JSR      PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
      .WORD    10 ;ERROR TYPE CODE.
      DEC      R3 ;DECREMENT 128 WORD COUNTER
      BNE     13$ ;BR IF MORE.
      DEC      R4 ;DECREMENT 4096. WORD COUNTER
      BNE     12$ ;BR IF MORE.
      BIT      R5, R2 ;CHECK FOR END OF A BLOCK.
      BNE     11$ ;BRANCH IF MORE IN CURRENT BLOCK.
      JSR      PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 11$.

```

F06

MAINDEC-11-DZQMC-8-0:
DZQMCB.P11 T20

0-124K MEMORY EXERCISER, 16K VER
8 XOR 13 TEST PATTERN

MACY11 27(732) 10-SEP-76 12:01 PAGE 71

3086
3087
3088
3089
3090
3091
3092
3093
3094
3095
3096
3097
3098
3099
3100
3101
3102
3103
3104
3105
3106
3107
3108
3109
3110
3111
3112
3113
3114
3115
3116
3117
3118
3119
3120
3121
3122
3123
3124
3125
3126
3127
3128
3129
3130
3131
3132
3133

012042 016746 167536
012046 012767 017777 167530
012054 004467 002650
012060 012704 004000
012064 005122
012066 005122
012070 005304
012072 001374
012074 030502
012076 001370
012100 004767 003402
012104 012667 167474

012110 012700 177777
012114 004467 002610
012120 012704 000040
012124 005100
012126 012703 000100
012132 005100
012134
012134 012201
012136 020001
012140 001405
012142 004767 006436
012146 004767 007664
012152 000010
012154
012154 012201
012156 020001
012160 001405
012162 004767 006416
012166 004767 007644
012172 000010
012174
012174 005303
012176 001356
012200 005304
012202 001351
012204 030502
012206 001344
012210 004767 003272

```
*****
* COMPLEMENT 8 XOR 13 TEST PATTERN.
*****
MOV     BLKMSK, -(SP) ;SAVE BLOCK MASK TEMPORARILY.
MOV     #MASK4K, BLKMSK ;SET BLOCK MASK TO 4K.
JSR     R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
21$:   MOV     #2048., R4 ;SET 4096. WORD COUNTER
22$:   COM     (R2)+ ;COMPLEMENT TEST PATTERN
        COM     (R2)+
        DEC     R4 ;COUNT 4K WORDS
        BNE    22$ ;BR IF MORE.
        BIT     R5, R2 ;CHECK FOR END OF A BLOCK.
        BNE    21$ ;BRANCH IF MORE IN CURRENT BLOCK.
        JSP    PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 21$.
        MOV     (SP)+, BLKMSK ;RESTORE BLOCK MASK.

*****
* CHECK COMPLEMENTED 8 XOR 13 TEST PATTERN WRITTEN ABOVE.
*****
MOV     #-1., R0 ;SET UP CHECK WORD.
JSR     R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
31$:   MOV     #32., R4 ;SET 4K WORD COUNTER
        COM     R0 ;COMPLEMENT TEST WORD
32$:   MOV     #64., R3 ;SET 128 WORD COUNTER
        COM     R0 ;COMPLEMENT TEST WORD
33$:   MOV     (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
        CMP     R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
        BEQ    69$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
68$:   JSR     PC, SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
        JSR     PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
        .WORD 10 ;ERROR TYPE CODE.
69$:   MOV     (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
        CMP     R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
        BEQ    71$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
70$:   JSR     PC, SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
        JSR     PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
        .WORD 10 ;ERROR TYPE CODE.
71$:   DEC     R3 ;DECREMENT 128 WORD COUNTER
        BNE    33$ ;BR IF MORE.
        DEC     R4 ;DECREMENT 4096. WORD COUNTER
        BNE    32$ ;BR IF MORE.
        BIT     R5, R2 ;CHECK FOR END OF A BLOCK.
        BNE    31$ ;BRANCH IF MORE IN CURRENT BLOCK.
        JSP    PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 31$.
```

```

3134
3135
3136
3137
3138
3139
3140
3141
3142
3143 012214 004567 006626
3144 012214 000000
3145 012220 012767 000021 166756
3146 012220 005767 170016
3147 012234 001404
3148 012236 032777 000100 166672
3149 012244 001402
3150 012246 000167 000606
3151 012252 005000
3152 012254 004767 004212
3153 012260 004467 002444
3154 012264 036767 167252 167244
3155 012272 001010
3156 012274 036767 167244 167236
3157 012302 001004
3158 012304 050502
3159 012306 005202
3160 012310 000167 000524
3161 012314 004767 005500
3162 012320 004767 005530
3163 012324 020227 000114
3164 012330 001004
3165 012332 062702 000004
3166 012336 000167 000476
3167 012342 111201
3168 012344 001405
3169 012346 004767 006156
3170 012352 004767 007460
3171 012356 000011
3172 012360
3173 012360 105067 167172
3174 012364 112700 000252
3175 012370 110012
3176 012372 016703 167220
3177 012376 052733 000005
3178 012402 005713
3179 012404 001374
3180
3181 012406 110012
3182 012410 016703 167202
3183 012414 042733 000004
3184 012420 005713
3185 012422 001374
3186 012424 016737 167176 000114
3187
3188 012432 105412
3189

```

```

*****
*TEST 21      WORSE CASE NOISE PARITY BYTE TESTING
* CHECK PARITY MEMORY WITH A SERIES OF BYTE PATTERNS
*   1) FORCE WRONG PARITY IN EACH BYTE OF PARITY MEMORY
*   2) READ IT BACK WITH ACTION ENABLE SET, MAKING SURE THAT A TRAP OCCURS
*   3) WRITE GOOD PARITY AND MAKE SURE NO TRAP OCCURS WHEN IT IS READ
*   4) MAKE SURE THE ERROR ADDRESS BITS (CSR BITS (11-5)) ARE CORRECT
*****
TST21:
      JSR      R5,      $SCOPE ;GO TO SCOPE ROUTINE.
      .WORD   0 ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
      MOV     #STN-1,$TESTN ;SET TEST NUMBER IN MAIL BOX
WWPB0: TST     MPRX ;CHECK FOR ANY PARITY MEMORY.
      BEQ    1$ ;BR IF NO PARITY MEMORY.
      BIT   #SW06, 2$WR ;CHECK FORINHIBIT PARITY SWITCH.
      BEQ    2$ ;BR IF NOT SET.
1$:   JMP     TST22 ;SKIP THIS TEST IF NO PARITY MEMORY PRESENT.
2$:   CLR     R0 ;ZERO TO BE PUT IN ALL MEMORY.
      JSR    PC,      SETCON ;ROUTINE TO LOAD ALL MEMORY.
      JSR    R4,      INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
WWPBYT: BIT   BITPT,  PMEMAP ;CHECK IF CURRENT BANK HAS PARITY MEMORY.
      BNE   2$ ;BR IF PARITY MEM.
      BIT   BITPT+2,PMEMAP+2 ;...HI 64K.
      BNE   2$ ;BR IF PARITY MEM.
      BIS   R5,      R2 ;POINT TO END OF BLOCK.
      INC   R2 ;FIRST ADR OF NEXT BLOCK.
      JMP   WWPBS ;BR TO FIND NEXT BLOCK.
2$:   JSR    PC,      SETAE ;SET ACTION ENABLE (EVEN IF BANK0.)
      JSR    PC,      CKPMER ;CHECK FOR ANY NON TRAP PARITY ERRORS.
WWPB1: CMP    R2,      #114 ;CKECK IF POINTING TO PARITY ERROR VECTOR.
      BNE   3$ ;BR IF NOT AT VECTOR.
      ADD   #4,      R2 ;SKIP PARITY VECTOR.
      JMP   WWPBS ;CHECK FOR BLOCK END.
3$:   MOVB   (R2),   R1 ;CHECK IF BYTE STILL CLEARED.
      BEQ   65$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
64$:  JSR    PC,      SPRNT ;SET UP VALUES FOR ERROR PRINTING.
      JSR    PC,      $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
      .WORD  11 ;ERROR TYPE CODE.
65$:  CLRB   OEFLG ;CLEAR ODD/EVEN FLAG.
      MOVB  #252,   R0 ;SET UP DATA...EVEN, SETS PARITY BIT.
WWPB2: MOVB   R0,      (R2) ;MOV DATA INTO TEST LOCATION.
      MOV   .MPRX,  R3 ;GET PARITY REGISTER TABLE POINTER.
10$:  BIS   #WWP+AE, 2(R3)+ ;SET WRITE WRONG PARITY.
      TST  (R3) ;CHECK FOR TABLE TERMINATOR.
      BNE  10$ ;BR IF MORE REGS IN TABLE.
* SET WRONG PARITY IN LOCATION UNDER TEST.
      MOVB  R0,      (R2) ;WRITE SAME DATA (EXCEPT PARITY) VIA DATOB.
      MOV   .MPRX,  R3 ;GET PARITY REG TABLE POINTER.
11$:  BIC   #WWP,    2(R3)+ ;CLEAR WRITE WRONG PARITY.
      TST  (R3) ;CHECK FOR TABLE TERMINATOR.
      BNE  11$ ;BR IF MORE PARITY REGISTERS.
* DETECT WRONG PARITY VIA DATIP; SET UP VECTOR FOR EXPECTED TRAP.
      MOV   .PBTRP, 2$PARVEC ;DATOB SHOULDN'T EXECUTE.
      NEGB (R2) ;DATIP (DATOB AND COM PARITY BIT.)
* SHOULD HAVE TRAPPED TO PBTRP.

```



```

3190 012434 016737 167172 000114      MOV      .PESRV, 0#PARVEC ;RESET VECTOR FOR UNEXPECTED TRAPS.
3191 012442 004767 006112      JSR      PC,      SPRNTO ;SET UP VALUES FOR ERROR PRINTING.
3192 012446 004767 007364      JSR      PC,      $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
3193 012452 000012      .WORD   12          ;ERROR TYPE CODE.
3194 012454 000557      BR      WWPB4      ;SKIP TRAP SERVICE.
3195
3196
3197 012456 016737 167150 000114  ;* EXPECTED PARITY MEMORY TRAPS COME HERE.
3198 012464 022626      PBTRP: MOV      .PESRV, 0#PARVEC ;RESET PARITY VECTOR FOR UNEXPECTED TRAPS.
3199 012466 016703 167122      CMP      (SP)+, (SP)+ ;RESET THE STACK POINTER AFTER TRAP.
3200 012472 032713 000001      MOV      .MPC, R3 ;GET PARITY REG AND MAP TABLE POINTER.
3201 012476 001003      21$: BIT      #BIT0, (R3) ;CHECK IF THIS REGISTER EXISTS.
3202 012500 017301 000000      BNE     22$ ;BR IF IT DOESN'T EXIST.
3203 012504 000413      MOV      2(R3), R1 ;GET THE CONTENTS.
3204 012506 062703 000006      BMI     23$ ;BR IF ERROR FLAG SET.
3205 012512 020367 167100      ADD      #6, R3 ;MOVE POINTER TO NEXT REG.
3206 012516 103765      CMP      R3, .MPX ;CHECK FOR END OF TABLE.
3207 012520 004767 006034      BLO     21$ ;BR IF MORE REGISTERS.
3208 012524 004767 007306      JSR      PC,      SPRNTO ;SET UP VALUES FOR ERROR PRINTING.
3209 012530 000013      JSR      PC,      $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
3210 012532 000530      .WORD   13          ;ERROR TYPE CODE.
3211 012534 036763 167002 000002 23$: BR      WWPB4      ;EXIT AFTER ERROR.
3212 012542 001011      BIT      BITPT, 2(R3) ;CHECK THE MAP FOR THIS REGISTER.
3213 012544 036763 166774 000004      BNE     24$ ;BR IF THIS REGISTER CONTROLS THIS BANK.
3214 012552 001005      BIT      BITPT+2, 4(R3) ;CHECK THE HI 64K.
3215 012554 004767 005774      BNE     24$ ;BR IF THIS REGISTER CONTROLS THIS BANK.
3216 012560 004767 007252      65$: JSR      PC,      SPRNTP ;SET UP VALUES FOR ERROR PRINTING.
3217 012564 000014      JSR      PC,      $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
3218 012566      .WORD   14          ;ERROR TYPE CODE.
3219
3220 012566 010046      24$: MOV      R0, -(SP) ;: PUSH R0 ON STACK
3221 012570 010200      MOV      R2, R0 ;: GET THE ADDRESS POINTER.
3222 012572 042700 003777      BIC      #3777, R0 ;: CLEAR LOW ADDRESS BITS.
3223 012576 000300      SWAB    R0 ;: SHIFT 6 PLACES RIGHT.
3224 012602 006300      ASL     R0
3225 012604 005767 165776      ASL     R0
3226 012610 001404      TST     MAVA ;CHECK FOR MEM MGMT.
3227 012612 042700 177600      BEQ     25$ ;BR IF NO MEM MGMT.
3228 012616 063700 172344      BIC      #177600, R0 ;CLEAR BANK BITS
3229 012622 052700 100001      ADD     0#KIPAR2, R0 ;ADD MEM MGMT OFFSET.
3230 012626 046700 166662      25$: BIS     #BIT15+BIT0, R0 ;SET ERROR AND AE BIT IN CHECK WORD.
3231 012632 046701 166656      BIC     RESRVD, R0 ;CLEAR PARITY REG BITS RESERVED FOR FUTURE.
3232      BIC     RESRVD, R1 ;CLEAR PARITY REG BITS RESERVED FOR FUTURE.
3233      ;NOTE: THE ABOVE INSTRUCTION (2 WORDS) CAN BE NOP'ED FOR UNMIXED MEMORY TYPES.
3234 012636 020001      CMP     R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
3235 012640 001405      BEQ     67$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
3236 012642 004767 005706      66$: JSR      PC,      SPRNTP ;SET UP VALUES FOR ERROR PRINTING.
3237 012646 004767 007164      JSR      PC,      $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
3238 012652 000015      .WORD   15          ;ERROR TYPE CODE.
3239
3240 012654 005073 000000      67$: CLR     2(R3) ;CLEAR REG INCLUDING ACTION ENABLE.
3241 012660 010346      MOV     R3, -(SP) ;: PUSH R3 ON STACK
3242 012662 062703 000006      26$: ADD     #6, R3 ;UPDATE POINTER TO NEXT PARITY REG + MAP.
3243 012666 020367 166724      CMP     R3, .MPX ;CHECK FOR END OF TABLE.
3244 012672 101014      BHI     WWPB3 ;BR IF END OF TABLE REACHED.
3245 012674 032713 000001      BIT     #BIT0, (R3) ;CHECK IF NEXT REG EXISTS.
3246 012700 001370      BNE     26$ ;BR IF THIS PARITY REG DOESN'T EXIST.

```

3246	012702	017301	000000		MOV	2(R3),	R1	;SAVE AND CHECK FOR ERROR FLAG.
3247	012706	100365			BPL	26\$;BR IF NO ERROR FLAG.
3248	012710	004767	005640	68\$:	JSR	PC,	SPRNT	;SET UP VALUES FOR ERROR PRINTING.
3249	012714	004767	007116		JSR	PC,	\$ERROR	;*** ERROR *** (GO TYPE A MESSAGE)
3250	012720	000016			.WORD	16		;ERROR TYPE CODE.
3251	012722	000757			BR	26\$;BR AFTER ERROR.
3252	012724	111204		WWPB3:	MOVB	(R2),	R4	;GET THE DATA FOR CHECKING.
3253					;* READING THE DATA VIA DATI TO CHECK IT SHOULD CAUSE PARITY ERROR, BUT			
3254					;* ACTION ENABLE IS NOT SET IN CONTROLLING REG, SO NO TRAP SHOULD OCCURE.			
3255	012726	111212			MOVB	(R2),	(R2)	;RESTORE RIGHT PARITY
3256				:NOTE:	THE ABOVE INSTRUCTION CAN BE NOP'ED FOR PROCESSORS			
3257				:	WHICH DC ONLY DATOB TO DESTINATION OF MOVB INSTRUCTIONS.			
3258	012730	012603			MOV	(SP)+,	R3	;POP STACK INTO R3
3259	012732	017301	000000		MOV	2(R3),	R1	;READ THE PARITY REGISTER TO CHECK IT AGAIN.
3260	012736	046701	166552		BIC	RESRVC,	R1	;CLEAR PARITY REG BITS RESERVED FOR FUTURE.
3261				:NOTE:	THE ABOVE INSTRUCTION (2 WORDS) CAN BE NOP'ED FOR JNMIXED MEMORY TYPES.			
3262	012742	042700	000001		BIC	#AE,	RO	;CLEAR THE ACTION ENABLE BIT IN TEST DATA.
3263	012746	020001			CMP	RO,	R1	;COMPARE THE CHECK WORD WITH THE DATA READ.
3264	012750	001405			BEQ	65\$;BRANCH OVER ERROR CALL IF GOOD DATA.
3265	012752	004767	005576	64\$:	JSR	PC,	SPRNT	;SET UP VALUES FOR ERROR PRINTING.
3266	012756	004767	007054		JSR	PC,	\$ERROR	;*** ERROR *** (GO TYPE A MESSAGE)
3267	012762	000015			.WORD	15		;ERROR TYPE CODE.
3268	012764			65\$:				
3269	012764	012773	000001	000000	MOV	#1,	2(R3)	;CLEAR ALL BUT ACTION ENABLE.
3270	012772	010401			MOV	R4,	R1	;GET DATA READ FROM MEMORY FOR TESTING.
3271	012774	012600			MOV	(SP)+,	RO	;POP STACK INTO RO
3272	012776	120001			CMPB	RO,	R1	;CHECK THE DATA.
3273	013000	001405			BEQ	67\$;BRANCH OVER ERROR CALL IF GOOD DATA.
3274	013002	004767	005552	66\$:	JSR	PC,	SPRNT	;SET UP VALUES FOR ERROR PRINTING.
3275	013006	004767	007024		JSR	PC,	\$ERROR	;*** ERROR *** (GO TYPE A MESSAGE)
3276	013012	000017			.WORD	17		;ERROR TYPE CODE.
3277	013014			67\$:				
3278	013014	110012		WWPB4:	MOVB	RO,	(R2)	;RESTORE DATA.
3279	013016	105712			TSTB	(R2)		;DO A DATI TO BE SURE RIGHT PARITY.
3280	013020	012700	000253		MOV	#253,	RO	;SET ODD PARITY DATA.
3281	013024	105167	166526		COMB	OEFLG		;CHECK IF DONE BOTH ODD AND EVEN PARITY.
3282	013030	100002			BPL	27\$;BR IF DONE BOTH EVEN AND ODD.
3283	013032	000167	177332		JMP	WWPB2		;LOOP BACK AND DO ODD(PARITY BIT CLR).
3284	013036	005202		27\$:	INC	R2		;MOVE POINTER TO NEXT MEMORY BYTE.
3285	013040	030502		WWPB5:	BIT	R5,	R2	;CHECK FOR END OF BLOCK.
3286	013042	001402			BEQ	30\$;BR IF END OF BLOCK FOUND.
3287	013044	000167	177254		JMP	WWPB1		;LOOP BACK TO TEST NEXT BYTE.
3288	013050	004767	002432	30\$:	JSR	PC,	MMUP	;FIND NEXT BLOCK AND LOOP TO WWPBYT
3289	013054	004767	004674		JSR	PC,	MAMF	;GO RESET PARITY REGISTERS.

JOB

MAINDEC-11-DZQMC-8-D:
DZQMCB.P11 T22

0-124K MEMORY EXERCISER, 16K VER
RANDOM DATA TESTING THRU PROGRAM CODE RELOCATION.

MACY11 27(732) 10-SEP-76 12:01 PAGE 75

```

3290                                     :*****
3291                                     :*TEST 22      RANDOM DATA TESTING THRU PROGRAM CODE RELOCATION.
3292                                     :*****
3293 013060                               †ST22:
3294 013060 004567 005762                JSR    R5,    $SCOPE ;GO TO SCOPE ROUTINE.
3295 013064 000000                        .WORD  0 ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
3296 013066 012767 000022 166112        MOV    #STN-1,$STSTN ;SET TEST NUMBER IN MAIL BOX
3297 013074 010703                        PC,    R3 ;GET CURRENT PROGRAM COUNTER.
3298 013076 042703 007777                BIC    #7777, R3 ;POINT TO BEGINNING OF CURRENT 2K BLOCK.
3299 013102 004467 001622                JSR    R4,    INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
3300 013106 010246                        1$:   MOV    R2,    -(SP) ;SAVE MEMORY POINTER.
3301 013110 010346                        MOV    R3,    -(SP) ;SAVE "DATA" POINTER.
3302 013112 012322                        2$:   MOV    (R3)+, (R2)+ ;MOV CODE INTO TEST MEMORY.
3303 013114 032703 007777                BIT    #7777, R3 ;CHECK FOR END OF "DATA TABLE"
3304 013120 001002                        BNE    3$ ;BRANCH IF MORE
3305 013122 162703 010000                SUB    #10000, R3 ;RESET POINTER TO START OF "RANDOM DATA"
3306 013126 030502                        3$:   BIT    R5,    R2 ;CHECK FOR END OF BLOCK
3307 013130 001370                        BNE    2$ ;BRANCH IF MORE.
3308 013132 012603                        MOV    (SP)+, R3 ;RESET "DATA" POINTER.
3309 013134 012602                        MOV    (SP)+, R2 ;RESET MEMORY POINTER.
3310 013136 012300                        4$:   MOV    (R3)+, R0 ;GET S/B DATA.
3311 013140 012201                        MOV    (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
3312 013142 020001                        CMP    R0,    R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
3313 013144 001405                        BEQ    65$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
3314 013146 004767 005432                64$:  JSR    PC,    SPRT2 ;SET UP VALUES FOR ERROR PRINTING.
3315 013152 004767 006660                JSR    PC,    $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
3316 013156 000020                        .WORD  20 ;ERROR TYPE CODE.
3317 013160                               65$:
3318 013160 032703 007777                BIT    #7777, R3 ;CHECK FOR END OF "DATA TABLE"
3319 013164 001002                        BNE    5$ ;BR IF MORE.
3320 013166 162703 010000                SUB    #10000, R3 ;RESET POINTER TO TOP OF "DATA TABLE".
3321 013172                               5$:
3322 013172 030502                        BIT    R5,    R2 ;CHECK FOR END OF A BLOCK.
3323 013174 001360                        BNE    4$ ;BRANCH IF MORE IN CURRENT BLOCK.
3324 013176 004767 002304                JSR    PC,    MMUP ;FIND NEXT BLOCK AND LOOP TO 1$.

```

3325
3326
3327
3328
3329
3330
3331
3332
3333
3334
3335
3336
3337
3338
3339
3340
3341
3342
3343
3344
3345
3346
3347
3348
3349
3350
3351
3352
3353
3354
3355
3356
3357
3358
3359
3360
3361
3362
3363
3364
3365
3366
3367
3368
3369
3370
3371
3372
3373
3374
3375

013202
013202 004567 005640
013206 000003
013210 000167 000064
013214 012767 000023 165764
013222 012703 010412
013226 012704 000205
013232 010400
013234 004467 001470
013240 010322
013242 010412
013244 004542
013246 012201
013250 020001
013252 001405
013254 004767 005320
013260 004767 006552
013264 000021
013266
013266 010322
013270 030502
013272 001363
013274 004767 002206

```

.SBTTL SECTION 3: INSTRUCTION EXECUTION TESTS.
*****
*TEST 23 EXECUTE DATI, DATO THRU MEMORY.
* EXECUTES THE INSTRUCTION 'MOV R4,(R2)' THROUGHOUT MEMORY.
* AN 'RTS R5' (CODE 205) IS PLACED AFTER THE 'MOV' INSTRUCTION TO RETURN
* CONTROL TO THE MAIN PROGRAM FOR INSTRUCTION EXECUTION CHECKOUT.
* THIS IS AN EXAMPLE OF WHAT THIS TEST DOES IN RELATION TO MEMORY:
*
* MEMORY LOCATION INSTRUCTION CONTENTS OF MEMORY LOCATION
* LOCATION PLACED THERE AFTER INSTRUCTION EXECUTION
*
* 1ST PASS / 40000 010412 000205
* THRU TEST / 40002 000205 000205
*
* 2ND PASS / 40002 010412 000205
* THRU TEST / 40004 000205 000205
*
* ETC., ETC., ETC.
*
* R0 = DATA WRITTEN ON TOP OF IUT BY THE IUT (SHOULD BE).
* R1 = DATA READ FROM MEMORY (WAS).
* R2 = ADDRESS OF IUT/DATA.
* R3 = INSTRUCTION UNDER TEST (IUT).
* R4 = RTS R5 (CODE 205).
* R5 = BLOCK BOUNDARY BIT MASK.
*****
TST23: JSR R5, $SCOPE ;GO TO SCOPE ROUTINE.
.WORD 3 ;MINIMUM BLOCK SIZE OF 2 WORDS
; REQUIRED FOR THIS TEST.
JMP TST24 ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
; AVAILABLE FOR TEST.
MOV #STN-1,$TESTN ;SET TEST NUMBER IN MAIL BOX
DIDO: MOV #010412,R3 ;GET 'MOV R4,(R2)' INSTRUCTION (IUT).
MOV #205,R4 ;GET 'RTS R5'
MOV R4, R0 ;SET UP S/B DATA AFTER EXECUTION.
JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$: MOV R3, (R2)+ ;PUT IUT INTO FIRST LOC OF BLOCK.
2$: MOV R4, (R2) ;PUT 'RTS R5' FOLLOWING IUT.
JSP R5, -(R2) ;GO EXECUTE THE IUT.
MOV (R2)+, R1 ;GET THE DATA FROM THE MEM ADR UNDER TEST.
CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
BEQ 65$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
64$: JSR PC, SPRNT3 ;SET UP VALUES FOR ERROR PRINTING.
JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
.WORD 21 ;ERROR TYPE CODE.
65$: MOV R3, (R2)+ ;PUT THE IUT INTO THE NEXT LOCATION.
BIT R5, R2 ;CHECK FOR END OF A BLOCK.
BNE 2$ ;BRANCH IF MORE IN CURRENT BLOCK.
JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 1$.

```

3376
3377
3378
3379
3380
3381
3382
3383
3384
3385
3386
3387
3388
3389
3390
3391
3392
3393
3394
3395
3396
3397
3398
3399
3400

```

*****
*TEST 24 EXECUTE DATI, DATOB (LOW BYTE) THRU MEMORY.
* EXECUTES THE INSTRUCTION 'MOVB R4,(R2)' THROUGHOUT MEMORY.
* AN 'RTS R5' (CODE 205) IS PLACED AFTER THE 'MOVB' INSTRUCTION TO RETURN
* CONTROL TO THE MAIN PROGRAM FOR INSTRUCTION EYECUTION CHECKOUT.
* THIS IS AN EXAMPLE OF WHAT THIS TEST DOES IN RELATION TO MEMORY:
*
*          MEMORY          INSTRUCTION          CONTENTS OF MEMORY LOCATION
*          LOCATION        PLACED THERE        AFTER INSTRUCTION EXECUTION
*
* 1ST PASS / 40000        110412             110605
* THRU TEST / 40002        000205             000205
*
* 2ND PASS / 40002        110412             110605
* THRU TEST / 40004        000205             000205
*
*          ETC.. ETC., ETC.
*
* R0 = DATA WRITTEN ON TOP OF IUT BY THE IUT (SHOULD BE).
* R1 = DATA READ FROM MEMORY (WAS).
* R2 = ADDRESS OF IUT/DATA.
* R3 = INSTRUCTION UNDER TEST (IUT).
* R4 = RTS R5 (CODE 205).
* R5 = BLOCK BOUNDARY BIT MASK.
*****

```

```

3401 013300
3402 013300 004567 005542
3403 013304 000003
3404
3405 013306 000167 000066
3406
3407 013312 012767 000024 165666
3408 013320 012703 110412
3409 013324 012704 000205
3410 013330 012700 110605
3411 013334 004467 001370
3412 013340 010322
3413 013342 010412
3414 013344 004542
3415 013346 012201
3416 013350 020001
3417 013352 001405
3418 013354 004767 005220
3419 013360 004767 006452
3420 013364 000021
3421 013366
3422 013366 010322
3423 013370 030502
3424 013372 001363
3425 013374 004767 002106

```

```

*****
TST24:
      JSR   R5,    $SCOPE ;GO TO SCOPE ROUTINE.
      .WORD 3        ;MINIMUM BLOCK SIZE OF 2 WORDS
                        ;REQUIRED FOR THIS TEST.
      JMP   TST25    ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                        ;AVAILABLE FOR TEST.
      MOV   #$STN-1,$TESTN ;SET TEST NUMBER IN MAIL BOX
      DIDBL: MOV   #110412,R3 ;GET 'MOVB R4,(R2)' INSTRUCTION (IUT).
      MOV   #205,R4 ;GET 'RTS R5'
      MOV   #110605,R0 ;SET UP S/B DATA AFTER EXECUTION.
      JSR   R4,    INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
      1$:  MOV   R3,    (R2)+ ;PUT IUT INTO FIRST LOC OF BLOCK.
      2$:  MOV   R4,    (R2) ;PUT 'RTS R5' FOLLOWING IUT.
      JSR   R5,    -(R2) ;GO EXECUTE THE IUT.
      MOV   (R2)+,  R1 ;GET THE DATA FROM THE MEM ADR UNDER TEST.
      CMP   R0,    R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
      BEQ   65$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
      64$:  JSR   PC,    SPRT3 ;SET UP VALUES FOR ERROR PRINTING.
      JSR   PC,    $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
      .WORD 21 ;ERROR TYPE CODE.
      65$:
      MOV   R3,    (R2)+ ;PUT THE IUT INTO THE NEXT LOCATION.
      BIT   R5,    R2 ;CHECK FOR END OF A BLOCK.
      BNE   2$ ;BRANCH IF MORE IN CURRENT BLOCK.
      JSR   PC,    MMUP ;FIND NEXT BLOCK AND LOOP TO 1$.

```

M06

MAINDEC-11-D2QMC-B-D:
D2QMCB.P11 T25

0-124K MEMORY EXERCISER, 16K VER
EXECUTE DATI, DATOB (HIGH BYTE) THRU MEMORY.

MACY11 27(732) 10-SEP-76 12:01 PAGE 78

3426
3427
3428
3429
3430
3431
3432
3433
3434
3435
3436
3437
3438
3439
3440
3441
3442
3443
3444
3445
3446
3447
3448
3449
3450
3451
3452
3453
3454
3455
3456
3457
3458
3459
3460
3461
3462
3463
3464
3465
3466
3467
3468
3469
3470
3471
3472
3473
3474
3475
3476

```

013400
013400 004567 005442
013404 000002
013406 000167 000072
013412 012767 000025 165566
013420 012703 110342
013424 012704 000205
013430 012700 161342
013434 004467 001270
013440 010322
013442 010412
013444 004562 177776
013450 005302
013452 012201
013454 020001
013456 001405
013460 004767 005114
013464 004767 006346
013470 000021
013472
013472 010322
013474 030502
013476 001361
013500 004767 002002
    
```

```

*****
*TEST 25 EXECUTE DATI, DATOB (HIGH BYTE) THRU MEMORY.
* EXECUTES THE INSTRUCTION 'MOVB R3, -(R2)' THROUGHOUT MEMORY.
* AN 'RTS R5' (CODE 205) IS PLACED AFTER THE 'MOVB' INSTRUCTION TO RETURN
* CONTROL TO THE MAIN PROGRAM FOR INSTRUCTION EXECUTION CHECKOUT.
* THIS IS AN EXAMPLE OF WHAT THIS TEST DOES IN RELATION TO MEMORY:
*
* MEMORY LOCATION INSTRUCTION CONTENTS OF MEMORY LOCATION
* LOCATION PLACED THERE AFTER INSTRUCTION EXECUTION
*
* 1ST PASS / 40000 110342 161342
* THRU TEST / 40002 000205 000205
*
* 2ND PASS / 40002 110342 161342
* THRU TEST / 40004 000205 000205
*
* ETC., ETC., ETC.
*
* R0 = DATA WRITTEN ON TOP OF IUT BY THE IUT (SHOULD BE).
* R1 = DATA READ FROM MEMORY (WAS).
* R2 = ADDRESS OF IUT/DATA.
* R3 = INSTRUCTION UNDER TEST (IUT).
* R4 = RTS R5 (CODE 205).
* R5 = BLOCK BOUNDARY BIT MASK.
*****
TST25:
JSR R5, $SCOPE ;GO TO SCOPE ROUTINE.
.WORD 3 ;MINIMUM BLOCK SIZE OF 2 WORDS
; REQUIRED FOR THIS TEST.
JMP TST26 ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
; AVAILABLE FOR TEST.
MOV #STN-1, $TESTN ;SET TEST NUMBER IN MAIL BOX
DIDBH: MOV #110342, R3 ;GET 'MOVB R3, -(R2)' INSTRUCTION (IUT).
MOV #205, R4 ;GET 'RTS R5'
MOV #161342, R0 ;SET UP S/B DATA AFTER EXECUTION.
JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$: MOV R3, (R2)+ ;PUT IUT INTO FIRST LOC OF BLOCK.
2$: MOV R4, (R2) ;PUT 'RTS R5' FOLLOWING IUT.
JSR R5, -2(R2) ;GO EXECUTE THE IUT.
DEC R2 ;ADJUST R2 TO POINT TO MAUT.
MOV (R2)+, R1 ;GET THE DATA FROM THE MEM ADR UNDER TEST.
CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
BEQ 65$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
34$: JSR PC, SPRINT3 ;SET UP VALUES FOR ERROR PRINTING.
JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
.WORD 21 ;ERROR TYPE CODE.
65$: MOV R3, (R2)+ ;PUT THE IUT INTO THE NEXT LOCATION.
BIT R5, R2 ;CHECK FOR END OF A BLOCK.
BNE 2$ ;BRANCH IF MORE IN CURRENT BLOCK.
JSR PC, MNUP ;FIND NEXT BLOCK AND LOOP TO 1$.
    
```

3477
3478
3479
3480
3481
3482
3483
3484
3485
3486
3487
3488
3489
3490
3491
3492
3493
3494
3495
3496
3497
3498
3499
3500
3501
3502
3503
3504
3505
3506
3507
3508
3509
3510
3511
3512
3513
3514
3515
3516
3517
3518
3519
3520
3521
3522
3523
3524
3525
3526

013504
013504 004567 005336
013510 000003

013512 000167 000066

013516 012767 000026 165462
013524 012703 005412
013530 012704 000205
013534 012700 172366
013540 004467 001164

013550 004542
013552 012201
013554 020001
013556 001405
013560 004767 005014
013564 004767 006246
013570 000021
013572
013572 010322
013574 030502
013576 001363
013600 004767 001702

```

*****
*TEST 26 EXECUTE DATI, DATIP, DATO THRU MEMORY.
* EXECUTES THE INSTRUCTION 'NEG (R2)' THROUGHOUT MEMORY.
* AN 'RTS R5' (CODE 205) IS PLACED AFTER THE 'NEG' INSTRUCTION TO RETURN
* CONTROL TO THE MAIN PROGRAM FOR INSTRUCTION EXECUTION CHECKOUT.
* THIS IS AN EXAMPLE OF WHAT THIS TEST DOES IN RELATION TO MEMORY:
*
* MEMORY LOCATION INSTRUCTION CONTENTS OF MEMORY LOCATION
* LOCATION PLACED THERE AFTER INSTRUCTION EXECUTION
*
* 1ST PASS / 40000 005412 172366
* THRU TEST / 40002 000205 000205
*
* 2ND PASS / 40002 005412 172366
* THRU TEST / 40004 000205 000205
*
* ETC., ETC., ETC.
*
* R0 = DATA WRITTEN ON TOP OF IUT BY THE IUT (SHOULD BE).
* R1 = DATA READ FROM MEMORY (WAS).
* R2 = ADDRESS OF IUT/DATA.
* R3 = INSTRUCTION UNDER TEST (IUT).
* R4 = RTS R5 (CODE 205).
* R5 = BLOCK BOUNDARY BIT MASK.
*****
TST26:
JSR R5, $SCOPE ;GO TO SCOPE ROUTINE.
.WORD 3 ;MINIMUM BLOCK SIZE OF 2 WORDS
;REQUIRED FOR THIS TEST.
JMP TST27 ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
;AVAILABLE FOR TEST.
MOV $STN-1,$TESTN ;SET TEST NUMBER IN MAIL BOX
DIPDO: MOV #005412,R3 ;GET 'NEG (R2)' INSTRUCTION (IUT).
MOV #205,R4 ;GET 'RTS R5'
MOV #172366,R0 ;SET UP S/B DATA AFTER EXECUTION.
JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$: MOV R3, (R2)+ ;PUT IUT INTO FIRST LOC OF BLOCK.
2$: MOV R4, (R2) ;PUT 'RTS R5' FOLLOWING IUT.
JSR R5, -(R2) ;GO EXECUTE THE IUT.
MOV (R2)+, R1 ;GET THE DATA FROM THE MEM ADR UNDER TEST.
CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
BEQ 65$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
64$: JSR PC, SPRT3 ;SET UP VALUES FOR ERROR PRINTING.
JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
.WORD 21 ;ERROR TYPE CODE.
65$: MOV R3, (R2)+ ;PUT THE IUT INTO THE NEXT LOCATION.
BIT R5, R2 ;CHECK FOR END OF A BLOCK.
BNE 2$ ;BRANCH IF MORE IN CURRENT BLOCK.
JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 1$.

```

000000
000001
000002
000003
000004
000005
000006
000007
000008
000009
000010
000011
000012
000013
000014
000015
000016
000017
000018
000019
000020
000021
000022
000023
000024
000025
000026
000027
000028
000029
000030
000031
000032
000033
000034
000035
000036
000037
000038
000039
000040
000041
000042
000043
000044
000045
000046
000047
000048
000049
000050
000051
000052
000053
000054
000055
000056
000057
000058
000059
000060
000061
000062
000063
000064
000065
000066
000067
000068
000069
000070
000071
000072
000073
000074
000075
000076
000077
000078
000079
000080
000081
000082
000083
000084
000085
000086
000087
000088
000089
000090
000091
000092
000093
000094
000095
000096
000097
000098
000099

```

*****
*TEST 27 EXECUTE DATI, DATI, DATIP, DATOB (LOW BYTE) THRU MEMORY.
* EXECUTES THE INSTRUCTION BICB (R2)+, -(R2) THROUGHOUT MEMORY.
* AN 'RTS R5' (CODE 205) IS PLACED AFTER THE BICB INSTRUCTION TO RETURN
* CONTROL TO THE MAIN PROGRAM FOR INSTRUCTION EXECUTION CHECKOUT.
* THIS IS AN EXAMPLE OF WHAT THIS TEST DOES IN RELATION TO MEMORY:
*
*          MEMORY LOCATION    INSTRUCTION    CONTENTS OF MEMORY LOCATION
*          LOCATION           PLACED THERE      AFTER INSTRUCTION EXECUTION
*
* 1ST PASS / 40000           142242         142000
* THRU TEST / 40002         000205         000205
*
* 2ND PASS / 40002           142242         142000
* THRU TEST / 40004         000205         000205
*
*      ETC., ETC., ETC.
*
* R0 = DATA WRITTEN ON TOP OF IUT BY THE IUT (SHOULD BE).
* R1 = DATA READ FROM MEMORY (WAS).
* R2 = ADDRESS OF IUT DATA.
* R3 = INSTRUCTION UNDER TEST (IUT).
* R4 = RTS R5 (CODE 205).
* R5 = BLOCK BOUNDARY BIT MASK.
*****
  
```

013604
013604 004567 005236
013610 000003

013612 000167 000066

013616 012767 000027 165362
013624 012703 142242
013630 012704 000205
013634 012700 142000
013640 004467 001064
013644 010322
013646 010412
013650 004542
013652 012201
013654 020001
013656 001405
013660 004767 004714
013664 004767 006146
013670 000021
013672
013672 010322
013674 030502
013676 001263
013700 004767 001602

```

†ST27:
JSR   R5,   $SCOPE ;GO TO SCOPE ROUTINE.
      .WORD 3       ;MINIMUM BLOCK SIZE OF 2 WORDS
                        ;REQUIRED FOR THIS TEST.
JMP   TST30 ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                        ;AVAILABLE FOR TEST.
MOV   #STN-1,STESTN ;SET TEST NUMBER IN MAIL BOX
DPDBL:MOV #142242,R3 ;GET BICB (R2)+, -(R2) INSTRUCTION (IUT).
      MOV #205, R4   ;SET 'RTS R5'
      MOV #142000,R0 ;SET UP S/B DATA AFTER EXECUTION.
      JSR R4,   INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
15:   MOV R3,   (R2)+ ;PUT IUT INTO FIRST LOC OF BLOCK.
25:   MOV R4,   (R2)  ;PUT 'RTS R5' FOLLOWING IUT.
      JSR R5,   -(R2) ;GO EXECUTE THE IUT.
      MOV (R2)+, R1   ;GET THE DATA FROM THE MEM ADR UNDER TEST.
      CMP R0,   R1   ;COMPARE THE CHECK WORD WITH THE DATA READ.
      BEQ 65$      ;BRANCH OVER ERROR CALL IF GOOD DATA.
64$:  JSR PC,   SPRNT3 ;SET UP VALUES FOR ERROR PRINTING.
      JSR PC,   $ERROR ;*** ERROR *** GO TYPE A MESSAGE.
      .WORD 21      ;ERROR TYPE CODE.
65$:  MOV R3,   R2)+ ;PUT THE IUT INTO THE NEXT LOCATION.
      BIT R5,   R2   ;CHECK FOR END OF A BLOCK.
      BNE 2$      ;BRANCH IF MORE IN CURRENT BLOCK.
      JSR PC,   MMUP ;FIND NEXT BLOCK AND LOOP TO 15.
  
```


35000
35001
35002
35003
35004
35005
35006
35007
35008
35009
35010
35011
35012
35013
35014
35015
35016
35017
35018
35019
35020
35021
35022
35023
35024
35025
35026
35027
35028
35029
35030

```
*****
*TEST 30 EXECUTE DATI, DATI, DATIP, DATOB (HIGH BYTE) THRU MEMORY.
* EXECUTES THE INSTRUCTION 'BISB (R2)+(R2)' THROUGHOUT MEMORY.
* AN 'RTS R5' (CODE 205) IS PLACED AFTER THE 'BISB' INSTRUCTION TO RETURN
* CONTROL TO THE MAIN PROGRAM FOR INSTRUCTION EXECUTION CHECKOUT.
* THIS IS AN EXAMPLE OF WHAT THIS TEST DOES IN RELATION TO MEMORY:
*
*      MEMORY LOCATION      INSTRUCTION      CONTENTS OF MEMORY LOCATION
*      LOCATION              PLACED THERE      AFTER INSTRUCTION EXECUTION
*
*      1ST PASS / 40000      152212           157212
*      THRU TEST / 40002      000205           000205
*
*      2ND PASS / 40002      152212           157212
*      THRU TEST / 40004      000205           000205
*
*      ETC., ETC., ETC.
*
* R0 = DATA WRITTEN ON TOP OF IUT BY THE IUT (SHOULD BE).
* R1 = DATA READ FROM MEMORY (WAS).
* R2 = ADDRESS OF IUT/DATA.
* R3 = INSTRUCTION UNDER TEST (IUT).
* R4 = RTS R5 (CODE 205).
* R5 = BLOCK BOUNDARY BIT MASK.
*****
```

013704	004557	005136
013704	000002	
013710	000002	
013712	000167	000070
013716	012767	000030
013724	012703	152212
013730	012704	000205
013734	012700	157212
013740	004467	000764
013744	010322	
013746	010412	
013750	004542	
013752	005302	
013754	012201	
013756	020001	
013760	001405	
013762	004767	004612
013766	004767	006044
013772	000021	
013774		
013774	010322	
013776	030502	
014000	001362	
014002	004767	001500

165262

```

TST30: JSR R5, $SCOPE :GO TO SCOPE ROUTINE.
        .WORD 3 :MINIMUM BLOCK SIZE OF 2 WORDS
                :REQUIRED FOR THIS TEST.
        JMP TST31 :SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                :AVAILABLE FOR TEST.
DPDB4: MOV $STN-1,$TESTN :SET TEST NUMBER IN MAIL BOX
        MOV $152212,R3 :GET 'BISB (R2)+(R2)' INSTRUCTION (IUT).
        MOV $205,R4 :GET 'RTS R5'
        MOV $157212,R0 :SET UP S/B DATA AFTER EXECUTION.
        JSR R4, INITMM :INITIALIZE THE MEMORY ADDRESS POINTERS.
        1$: MOV R3, (R2)+ :PUT IUT INTO FIRST LOC OF BLOCK.
        2$: MOV R4, (R2) :PLT 'RTS R5' FOLLOWING IUT.
        JSR R5, -(R2) :GO EXECUTE THE IUT.
        DEC R2 :RESET R2 TO POINT TO IUT.
        MOV (R2)+, R1 :GET THE DATA FROM THE MEM ADR UNDER TEST.
        CMP R0, R1 :COMPARE THE CHECK WORD WITH THE DATA READ.
        BEQ 65$ :BRANCH OVER ERROR CALL IF GOOD DATA.
        64$: JSR PC, $PRNT3 :SET UP VALUES FOR ERROR PRINTING.
        JSR PC, $ERROR :*** ERROR *** (GC TYPE A MESSAGE)
        .WORD 21 :ERROR TYPE CODE.
        65$: MOV R3, (R2)+ :PUT THE IUT INTO THE NEXT LOCATION.
        BIT R5, R2 :CHECK FOR END OF A BLOCK.
        BNE 2$ :BRANCH IF MORE IN CURRENT BLOCK.
        JSR PC, $MJP :FIND NEXT BLOCK AND LOOP TO 1$.
  
```

```

3628
3629
3630
3631
3632
3633
3634
3635
3636
3637
3638
3639 014006
3640 014006 004567 005034
3641 014012 000077
3642
3643 014014 000167 000326
3644
3645 014020 012767 000031 165160
3646 014026
3647 014026 004467 000676
3648 014032 012746 000024
3649 014036 010246
3650 014040 016700 165554
3651 014044 012701 000025
3652 014050 012022
3653 014052 030502
3654 014054 001404
3655 014056 005301
3656 014060 001373
3657 014062 005742
3658 014064 000765
3659 014066 005301
3660 014070 001407
3661 014072 162701 000025
3662 014076 005042
3663 014100 005201
3664 014102 001375
3665 014104 016722 000064
3666 014110 011602
3667 014112 016700 165504
3668 014116 016701 165502
3669 014122 000112
3670
3671
3672
3673 014124 010703
3674 014126 062703 000024
3675 014132 010304
3676 014134 005204
3677 014136 005013
3678 014140 000261
3679 014142 105513
3680 014144 100403
3681 014146 105214
3682 014150 000773
3683 014152 000

```

```

*****
:*TEST 31 "BRANCH GOBBLE" TEST
:* THIS TEST LOADS THE ROUTINE FOUND AT "BRGOB" BELOW INTO THE BEGINNING
:* OF MEMORY UNDER TEST. IF THERE IS ROOM, IT LOADS THE ROUTINE AGAIN
:* STARTING AT THE LAST INSTRUCTION OF THE FIRST COPY, THUS OVERLAYING
:* THE EXIT INSTRUCTION (JMP (R1)) OF THE FIRST COPY. THIS CONTINUES
:* UNTIL THE CURRENT BANK(S) ARE FULL. THE CODE IS THEN EXECUTED.
:* THIS ENTIRE PROCESS IS THEN REPEATED STARTING AT THE SECOND ADDRESS
:* OF THE BANK UNDER TEST, THEN THE THIRD, ETC., UNTIL EACH INSTRUCTION
:* OF THE ROUTINE HAS BEEN LOADED INTO AND EXECUTED OUT OF EVERY LOCATION.
*****
†TST31:
      JSR      R5,      $SCOPE ;GO TO SCOPE ROUTINE.
      .WORD   77
      JMP      TST32
      MOV      #STN-1,$TESTN ;;SET TEST NUMBER IN MAIL BOX
BRGOBL: JSR      R4,      INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
BGTOP:  MOV      #BGLN-1, -(SP) ;SAVE ROUTINE LENGTH
BGLTOP: MOV      R2,      -(SP) ;SAVE 1ST BLOCK ADDRESS.
5$:     MOV      .BRGOB, R0 ;GET FIRST LOC OF "BRANCH GOBBLE" ROUTINE.
      MOV      #BGLN, R1 ;SET UP LENGTH COUNTER
6$:     MOV      (R0)+, (R2)+ ;MOV ROUTINE INTO MEMORY
      BIT      R5,      R2 ;CHECK FOR END OF BLOCK.
      BEQ     7$
      DEC     R1 ;COUNT WORDS IN ROUTINE.
      BNE     5$ ;BR IF MORE.
      TST     -(R2) ;BACK UP ONE TO OVERLAY RETURN.
      BR     5$ ;BR TO START ANOTHER ROUTINE.
7$:     DEC     R1 ;CHECK FOR EXACT FINISH.
      BEQ     9$ ;BR IF ALL OK.
      SUB     #BGLN, R1 ;SET COUNTER FOR BACKING UP.
8$:     CLR     -(R2) ;PUT HALTS IN EXTRA MEMORY.
      INC     R1 ;COUNT NUMBER OF WORDS TO BACK UP..
      BNE     8$ ;BR IF MORE.
      MOV     BGEND, (R2)+ ;INSERT RETURN INSTRUCTION.
9$:     MOV     (SP), R2 ;RESET POINTER TO BEGINNING OF BLOCK.
      MOV     .BGERR, R0 ;POINT TO ERROR ROUTINE.
      MOV     .BGEXI, R1 ;POINT TO EXIT ROUTINE.
      JMP     (R2) ;GO TO BRANCH GOBBLE ROUTINES IN MEMORY.

:* "BRANCH GOBBLE" ROUTINE WHICH IS RELOCATED INTO MEMORY UNDER TEST
:* AND THEN EXECUTED THERE.
BRGCB: MOV      PC,      R3 ;GET CURRENT ROUTINE LOCATION.
BG1:   ADD     #BG3-9C1, R3 ;POINT TO DATA WORD "BG3"
      MOV     R3,      R4 ;COPY
      INC     R4 ;POINT TO DATA BYTE "BG4"
      CLR     (R3) ;CLR DATA WORD "BG3" & "BG4"
BG2:   SEC ;SET CARRY...TO BE ADDED TO "BG3"
      ADCB   (R3) ;ADD THE CARRY... AS IF INC INS
      BMI   BG5 ;BRANCH WHEN BIT 7 IS SET
      INCB   (R4)
      BR     BG2
BG3:   .BYTE   0 ;DATA BYTE POINTED TO BY R3

```

3684	014153	000			BG4:	.BYTE	0				:DATA BYTE POINTED TO BY R4
3685	014154	102401			BG5:	BVS	BG5				:BRANCH IF V-BIT SET...IT SHOULD BE.
3686	014156	004510				JSR	R5.	(R0)			:ERROR!!! V-BIT NOT SET.
3687	014160	000242			BG6:	CLV					:CLR V-BIT
3688	014162	105214				INCB	(R4)				:ONE MORE INC ON "BG4"
3689	014164	103402				BCS	BG7				:BR IF C SET...IT SHOULD NOT BE.
3690	014166	102001				SVC	BG7				:BR IF V SET...IT SHOULD NOT BE.
3691	014170	100401				BMI	BGEND				:BR IF N SET...IT SHOULD BE.
3692	014172	004510			BG7:	JSR	R5	(R0)			:ERROR!!! CC NOT = 2
3693	014174	000111			BGEND:	JMP	(R1)				:EXIT CODE...ONLY AT END OF BLOCK.
3694		000025			BGLEN=	<.-BRGOB>/2					:CALCULATED LENGTH OF ROUTINE IN WORDS.
3695											
3696	014176	013767	177776	164760	BGERR:	MOV	#PSW, \$TMP3				:SAVE PSW
3697	014204	010567	164746			MOV	R5, \$TMP0				:SAVE ERROR PC
3698	014210	160367	164742			SUB	R3, \$TMP0				:OFFSET IT TO THE DATA POINTER
3699	014214	062767	014150	164734		ADD	#BG3-2, \$TMP0				:MAKE IT VIRTUAL SO IT CAN BE FOUND IN LISTING
3700	014222	010567	164732			MOV	R5, \$TMP1				:GET ERROR PC
3701	014226	162767	000002	164724		SUB	#2, \$TMP1				:MAKE IT PHYSICAL
3702	014234	012767	000012	164720		MOV	#12, \$TMP2				:SET UP S/B PSW
3703	014242	010367	164652			MOV	R3, \$GADDR				:GET PHYSICAL DATA LOCATION
3704	014246	011367	164654			MOV	(R3), \$BDDAT				:GET THE DATA
3705	014252	026727	164700	014172		CMP	\$TMP0, #BG7				:CHECK WHICH ERROR
3706	014260	001404				BEQ	1\$:BRANCH IF SECOND ERROR
3707	014262	012767	077600	164634		MOV	#077600, \$GDDAT				:SET UP S/B DATA
3708	014270	000403				BR	2\$:SKIP
3709	014272	012767	100200	164624	1\$:	MOV	#100200, \$GDDAT				:SET UP S/B DATA
3710	014300				2\$:						
3711	014300	004767	005532			JSR	PC, \$ERROR				:*** ERROR *** (GO TYPE A MESSAGE)
3712	014304	000022				.WORD	22				:ERROR TYPE CODE.
3713	014306	000205				RTS	R5				:CONTINUE TESTING.
3714											
3715											
3716	014310	012602									
3717	014312	005722									
3718	014314	005767	164670								
3719	014320	001402									
3720	014322	005316									
3721	014324	001244									
3722	014326	005726									
3723	014330	050502									
3724	014332	005202									
3725	014334	004767	001146								

: * BRANCH GOBBLE EXIT ROUTINE
 BGEXIT: MOV (SP)+, R2 :GET FIRST ADDRESS OF BLOCK (SHIFTED).
 TST (R2)+ :SHIFT IT ONE WORD.
 TST \$PASS :CHECK FOR PASS 0.
 BEQ 1\$:BR IF FIRST PASS
 DEC (SP) :COUNT TIMES SHIFTED
 BNE BGLoop :BR IF NOT DONE ENOUGH SHIFTS.
 TST (SP)+ :ADJUST STACK POINTER.
 BVS R5, R2 :RESET ADR POINTER TO TOP OF BLOCK.
 INC R2 :GO TO NEXT BLOCK
 JSR PC, MMUP :GO FIND NEXT BLOCK AND LOOP TO BGLoop.

```

SBTTL DONE: RELOCATE PROGRAM AND REPEAT ALL TESTS.
DONE:
014340 004567 004502 JSR RS, $SCOPE ;GO TO SCOPE ROUTINE.
014340 000000 .WORD 0 ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
014344 005067 164614 TST32: CLR $TIMES ;RESET ITERATION COUNTER FOR RESTARTING TEST.
014346 1C5067 164524 CLRB $TSTNM ;RESET TEST NUMBER.
014352 036767 164220 165146 1$: BIT PRGMAP, SAVTST ;CHECK IF PROGRAM IS IN TEST AREA.
014356 001004 2$ ;BR IF IT PROG IN MEM TO BE TESTED.
014364 036767 164212 165140 BIT PRGMAP+2, SAVTST+2 ;CHECK HI 64K
014374 001426 BEQ $EOP ;BR IF PROG NOT IN MEM TO BE TESTED.
014376 032777 000200 164532 2$: BIT $SW07, $SWR ;CHECK FOR INHIBIT RELOCATION SWITCH.
014404 001022 BNE $EOP ;SKIP RELOCATION IF SWITCH SET.
014406 022767 000003 164156 CMP #3, PRGMAP ;CHECK IF PROGRAM IN FIRST BK.
014414 001007 BNE 4$ ;BR IF NOT IN FIRST BK.
014416 005737 000042 TST 2#42 ;CHECK FOR A MONITOR.
014422 001006 BNE 5$ ;BR IF A MONITOR.
014424 004767 002332 JSR PC, RELTOP ;RELOCATE PROGRAM TO TOP OF MEMORY.
014430 000167 170730 3$: JMP START1 ;LOOP BACK AND RUN ALL TESTS AGAIN.

4$: JSR PC, RELO ;RELOCATE PROGRAM BACK TO FIRST BK.
5$: JSR PC, RESLDR ;RESTORE LOADERS.
JSR RS, $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
.WORD $CALF ;ADDRESS OF MESSAGE TO BE TYPED

```

```

3749
3750
3751
3752
3753
3754
3755
3756
3757
3758
3759
3760
3761
3762
3763
3764
3765
3766
3767
3768
3769
3770
3771
3772
3773
3774
3775
3776
3777
3778
3779
3780
3781
3782
3783
3784
3785
3786
3787
3788
3789
3790
3791
3792
014453
014454
014454
014454
014460
014464
014472
014474
014476
014500
014502
014504
014506
014512
014514
014520
014524
014530
014534
014536
014536
014542
014544
014546
014550
014552
014554
014556
014556
014562
014570
014576
014577
000240
005067 164506
005267 164524
042767 100000 164516
005327
000001
003027
012737
000001
014474
004567 006526
014562
016746 164470
013746 177776
004767 007424
004567 006504
014577
016700 163300
001405
000005
004710
000240
000240
000240
000167 170602
005015 047105 020104
040520 051523 021440
000
377 000

```

```

:*****
.SBTTL END OF PASS ROUTINE

:*INCREMENT THE PASS NUMBER ($PASS)
:*TYPE "END PASS #XXXXX" (WHERE XXXXX IS A DECIMAL NUMBER)
:*IF THERES A MONITOR GO TO IT
:*IF THERE ISN'T JUMP TO START!

$EOP:
NOP
CLR $TIMES ;;ZERO THE NUMBER OF ITERATIONS
INC $PASS ;;INCREMENT THE PASS NUMBER
BIC #100000,$PASS ;;DON'T ALLOW A NEG. NUMBER
DEC (PC)+ ;;LOOP?

$EOPCT: .WORD 1
BGT $DOAGN ;;YES
MOV (PC)+,(PC)+ ;;RESTORE COUNTER

$ENDCT: .WORD 1

JSR RS, $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
.WORD $ENDMG ;ADDRESS OF MESSAGE TO BE TYPED
MOV $PASS, -(SP) ;SAVE $PASS FOR TYPEOUT
:* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPDS ROUTINE
:* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSTEMAC**.
MOV $PSW, -(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
JSR PC, $TYPDS ;GO TO THE SUBROUTINE
JSR RS, $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
.WORD $ENULL ;ADDRESS OF MESSAGE TO BE TYPED

$GET42:
MOV 42, RO ;;GET MONITOR ADDRESS
BEQ $DOAGN ;;BRANCH IF NO MONITOR
RESET ;;CLEAR THE WORLD
$ENDAD: JSR PC, (RO) ;;GO TO MONITOR
NOP ;;SAVE ROOM
NOP ;;FOR
NOP ;;ACT11

$DOAGN:
JMP START1 ;;RETURN*****
$ENDMG: .ASCIZ (15)(12)/END PASS #/

$ENULL: .BYTE -1,-1,0 ;;NULL CHARACTER STRING

```

3793
3794
3795
3796
3797
3798
3799
3800
3801
3802
3803
3804
3805
3806
3807
3808
3809
3810
3811
3812
3813
3814
3815
3816
3817
3818
3819
3820
3821
3822
3823
3824
3825
3826
3827
3828
3829
3830
3831
3832
3833
3834
3835
3836
3837
3838
3839
3840
3841
3842
3843
3844
3845
3846
3847
3848

014602
014602 012737 077406 172300
014610 012737 077406 172302
014616 012737 077406 172304
014624 012737 077406 172306
014632 005037 172310
014636 005037 172312
014642 005037 172314
014646 012737 077406 172316
014654 005037 172340
014660 012737 000200 172342
014666 005037 172344
014672 005037 172346
014676 005037 172350
014702 005037 172352
014706 005037 172354
014712 012737 007600 172356
014720 012737 000001 177572
014726 000207

```
.SBTTL SUBROUTINE AND TRAP ROUTINE SECTION.
.SBTTL MEMORY MANAGEMENT AND ADDRESSING SUBROUTINES.
*****
* SET UP ALL THE MEM MGMT REGISTERS FOR NORMAL OPERATION.
* THE PROGRAM IS POINTED TO BY PARS 0 AND 1.
* THE MEMORY UNDER TEST IS POINTED TO BY PARS 2 AND 3.
* THE DEVICE ADDRESS AREA IS POINTED TO BY PAR 7.
* PARS 4, 5, AND 6 ARE UNUSED.
*****
MMINIT:
MOV      #200-1*400+UP+RW, @#KIPDR0      ;SET KIPDR0 = RW UP 200 BLOCKS
MOV      #200-1*400+UP+RW, @#KIPDR1      ;SET KIPDR1 = RW UP 200 BLOCKS
MOV      #200-1*400+UP+RW, @#KIPDR2      ;SET KIPDR2 = RW UP 200 BLOCKS
MOV      #200-1*400+UP+RW, @#KIPDR3      ;SET KIPDR3 = RW UP 200 BLOCKS
CLR      @#KIPDR4
CLR      @#KIPDR5
CLR      @#KIPDR6
MOV      #200-1*400+UP+RW, @#KIPDR7      ;SET KIPDR7 = RW UP 200 BLOCKS
CLR      @#KIPAR0      ;MAP PAR0 INTO BANK0
MOV      #200, @#KIPAR1      ;MAP PAR1 INTO BANK1
CLR      @#KIPAR2      ;MAP PAR2 INTO BANK0
CLR      @#KIPAR3
CLR      @#KIPAR4
CLR      @#KIPAR5
CLR      @#KIPAR6
MOV      #7600, @#KIPAR7 ;MAP PAR7 INTO I/O BANK
MOV      #1, @#SRO      ;ENABLE MEMORY MANAGEMENT
RTS      PC              ;RETURN
```

```
*****
* MEMORY ADDRESS POINTER INITIALIZATION ROUTINES.
*****
INITMM: MOV      #BIT0, BITPT      ;SET POINTER TO BANK0
CLR      BITPT+2      ;CLEAR HI 64K BANK POINTERS
CLR      R2            ;SET ADDRESS POINTER TO 0
MOV      BLKMSK, R5    ;RESET R5 TO BLOCK MASK.
TST      MMAVA        ;CHECK FOR MEM MGMT AVAILABLE
BEQ      10$          ;BRANCH IF NO MEM MGMT
CLR      @#KIPAR2      ;SET UP 3RD PAR TO BANK0
MOV      #40000, R2    ;RESET VIRTUAL ADR POINTER
BIT      BITPT, TSTMAP ;CHECK IF THIS BANK TO BE TESTED
BNE      2$           ;BRANCH IF MATCH
BIT      BITPT+2, TSTMAP+2 ;CHECK IN HI MAP
BNE      2$           ;BRANCH IF MATCH
ADD      #200, @#KIPAR2 ;UPDATE MEM MGMT, THIRD PAR.
ASL      BITPT        ;UPDATE LO POINTER TO NEXT BANK.
ROL      BITPT+2      ;...HI POINTER.
BPL      1$           ;BR IF MORE.
HALT     ;FATAL ERROR!!! NO 4K BANK FOUND
BIT      BITPT, LADMAP ;CHECK IF LAST BANK.
BNE      3$           ;BR IF LAST BANK.
BIT      BITPT+2, LADMAP+2 ;CHECK IF LAST BANK.
BEQ      4$           ;BR IF NOT LAST BANK.
MOV      LADMSK, R5    ;SET MASK TO FIND LAST ADR.
BIC      #20000, TEMPLAD ;MAKE SJRE VIRTJAL LAST ADR IN BANK 2.
```

3849	015062	013737	172344	172346	4\$:	MOV	2#KIPAR2,2#KIPAR3	:COPY CURRENT PAR INTO FORTH PAR.
3850	015070	016757	164446	164450		MOV	BITPT, TMPPT	:COPY BITPT...LO 64K.
3851	015076	016767	164442	164444		MOV	BITPT+2, TMPPT+2	:...HI 64K.
3852	015104	032705	020000			BIT	#BIT13, R5	:CHECK FOR A BLOCK SIZE OF 8K.
3853	015110	001505				BEQ	21\$:BRANCH IF NOT 8K.
3854	015112	062737	000200	172346	5\$:	ADD	#200, 2#KIPAR3	:UP DATE FORTH PAR.
3855	015120	006367	164422			ASL	TMPPT	:UPDATE LO POINTER TO NEXT 4K BANK.
3856	015124	006167	164420			ROL	TMPPT+2	:...HI POINTER.
3857	015130	100473				BMI	20\$:BR IF NO MORE.
3858	015132	036767	164410	164366		BIT	TMPPT, TSTMAP	:CHECK IF BANK TO BE TESTED.
3859	015140	001004				BNE	6\$:BRANCH IF A MATCH.
3860	015142	036767	164402	164360		BIT	TMPPT+2, TSTMAP+2	:CHECK FOR HI 64K BANKS.
3861	015150	001760				BEQ	5\$:BRANCH IF NO MEMORY
3862	015152	036767	164370	164420	5\$:	BIT	TMPPT, LADMAP	:CHECK IF LAST BANK.
3863	015160	001004				BNE	7\$:BRANCH IF A MATCH
3864	015162	036767	164362	164412		BIT	TMPPT+2, LADMAP+2	:CHECK HI 64K
3865	015170	001455				BEQ	21\$:BR IF NOT LAST BANK.
3866	015172	016705	164400		7\$:	MOV	LADMSK, R5	:RESET MASK TO FIND LAST ADR.
3867	015176	052767	020000	164370		BIS	#20000, TEMPLAD	:MAKE SURE LAST ADDRESS IS IN BANK 3.
3868	015204	000447				BR	21\$:BR TO FINISH UP.
3869								
3870	015206	036767	164330	164312	10\$:	BIT	BITPT, TSTMAP	:CHECK IF THIS BANK TO BE TESTED.
3871	015214	001006				BNE	11\$:BR IF MATCH.
3872	015216	062702	020000			ADD	#20000, R2	:UPDATE PHYSICAL ADR PNTR TO NEXT BANK.
3873	015222	106367	164314			ASLB	BITPT	:UPDATE BANK POINTER TO NEXT BANK.
3874	015226	100367				BPL	10\$:BR IF MORE BANKS.
3875	015230	000000				HALT		:FATAL ERROR!!! NO 4K BANK FOUND?
3876	015232	016767	164304	164306	11\$:	MOV	BITPT, TMPPT	:COPY BANK POINTER.
3877	015240	036767	164276	164332		BIT	BITPT, LADMAP	:CHECK IF LAST BANK.
3878	015246	001021				BNE	12\$:BR IF LAST BANK.
3879	015250	032705	020000			BIT	#BIT13, R5	:CHECK FOR 8K BLOCK SIZE.
3880	015254	001423				BEQ	21\$:BRANCH IF SMALLER BLOCK SIZE.
3881	015256	106367	164264			ASLB	TMPPT	:POINT TO NEXT BANK.
3882	015262	100416				BMI	20\$:BRANCH IF OVERFLOW.
3883	015264	036767	164256	164234		BIT	TMPPT, TSTMAP	:CHECK IF BANK TO BE TESTED.
3884	015272	001412				BEQ	20\$:BRANCH IF NOT TO BE TESTED.
3885	015274	112767	000011	164253		MOV	#11, FLAG8K	:SET 8K BLOCK SIZE FLAG.
3886	015302	036767	164240	164270		BIT	TMPPT, LADMAP	:CHECK FOR LAST BANK.
3887	015310	001403				BEQ	20\$:BR IF NOT LAST BANK.
3888	015312	016705	164260		12\$:	MOV	LADMSK, R5	:RESET MASK TO FIND LAST ADR.
3889	015316	000402				BR	21\$:SKIP MASK RESET.
3890	015320	012705	017777		20\$:	MOV	#MASK4K, R5	:RESET MASK TO 4K BLOCK SIZE.
3891	015324	056767	164212	164214	21\$:	BIS	BITPT, TMPPT	:SET TMPPT FOR FLAGING LAST BANK.
3892	015332	056767	164206	164210		BIS	BITPT+2, TMPPT+2	
3893	015340	036767	164176	164220		BIT	BITPT, FADMAP	:CHECK IF FIRST ADDRESS NEEDS TO BE SET.
3894	015346	001004				BNE	22\$:BR IF FIRST BANK.
3895	015350	036767	164170	164212		BIT	BITPT+2, FADMAP+2	:CHECK HI 64K.
3896	015356	001450				BEQ	INITEX	:BR IF NOT FIRST BANK.
3897	015360	016702	164176		22\$:	MOV	TMPPAD, R2	:RESET ADDRESS POINTER TO FIRST ADR.
3898	015364	000445				BR	INITEX	
3899								
3900	015366	016705	164212		INITDN:	MOV	BLKMSK, R5	:RESET R5 TO CURRENT BLOCK MASK.
3901	015372	005002				CLR	R2	:INIT ADDRESS POINTER.
3902	015374	005767	163206			TST	MMAVA	:CHECK FOR MEM MGMT
3903	015400	001411				BEQ	31\$:BRANCH IF NO MEM MGMT
3904	015402	012767	100000	164134		MOV	#BIT15, BITPT+2	:SET POINTER TO TOP BIT

```

3905 015410 005067 164126 CLR BITPT
3906 015414 012737 007600 172344 MOV #7600, 2#KIPAR2 ;SET PAR TO TOP OF MEM
3907 015422 000403 BR 32$ ;BRANCH TO COMMON AREA
3908
3909 015424 012767 000400 164110 31$: MOV #BIT8, BITPT ;SET UP BANK POINTER
3910 015432 012767 015454 164112 32$: MOV #33$, MMORE ;SET "MMDOWN" EXIT ADDRESS.
3911 015440 066767 163134 164104 ADD RELOCF, MMORE ;ADD OFFSET
3912 015446 004767 000524 JSR PC, MMDCWN ;ROUTINE TO SEARCH DOWNWARD FOR TOP MEM BANK
3913 015452 000000 HALT ;FATAL ERROR!!! NO MEM INDICATED IN MEM MAP ABOVE BK!
3914 015454 036767 164062 164116 33$: BIT BITPT, LADMAP ;CHECK FOR NON BOUNDARY LAST ADDR.
3915 015462 001004 BNE 34$ ;BR IF LAST BANK FLAG FOUND.
3916 015464 036767 164054 164110 BIT BITPT+2, LADMAP+2 ;CHECK FOR NON BOUNDARY LAST ADDR.
3917 015472 001402 BEQ INITEX ;BR IF NO LAD FLG FOUND.
3918 015474 016702 164072 34$: MOV LSTADR, R2 ;SET UP R2.
3919 015500 010467 164046 INITEX: MOV R4, MMORE ;PUT RETURN PC INTO "MMORE"
3920 015504 000204 RTS R4 ;RETURN
3921
3922 ;*****
3923 ;* COMMON UPWARDS ADDRESSING ROUTINE
3924 ;* FINDS NEXT EXISTING 4K BANK AND UPDATES POINTERS.
3925 ;* GOES TO ADDRESS IN "MMORE" IF MORE BANKS.
3926 ;* DOES STRAIGHT EXIT WHEN ALL MEMORY HAS BEEN DONE.
3927 ;*****
3928 015506 036767 164034 164064 MMJP: BIT TMPPT, LADMAP ;CHECK FOR LAST BANK FLAG.
3929 015514 001122 BNE 10$ ;BR IF LAST BANK.
3930 015516 036767 164026 164056 BIT TMPPT+2, LADMAP+2 ;CHECK FOR LAST BANK FLAG.
3931 015524 001116 BNE 10$ ;BR IF LAST BANK.
3932 015526 016705 164052 MOV BLKMSK, R5 ;RESET R5 TO BLOCK MASK.
3933 015532 005767 163050 TST MAVA ;CHECK FOR MEM MGMT AVAILABLE
3934 015536 001515 BEQ 20$ ;BRANCH IF NO MEM MGMT
3935 015540 012702 040000 MOV #40000, R2 ;RESET VIRTUPL ADR POINTER
3936 015544 062737 000200 172344 1$: ADD #200, 2#KIPAR2 ;UPDATE MEM MGMT, THIRD PAR.
3937 015552 006367 163764 ASL BITPT ;UPDATE LO POINTER TO NEXT BANK.
3938 015556 006167 163762 ROL BITPT+2 ;...HI POINTER.
3939 015562 100577 BMI 32$ ;BR IF ALL DONE.
3940 015564 036767 163752 163734 BIT BITPT, TSTMAP ;CHECK IF THIS BANK EXISTS
3941 015572 001004 BNE 2$ ;BRANCH IF MATCH
3942 015574 036767 163744 163726 BIT BITPT+2, TSTMAP+2 ;CHECK IN HI MAP
3943 015602 001760 BEQ 1$ ;BRANCH IF NO MATCH
3944 015604 036767 163732 163766 2$: BIT BITPT, LADMAP ;CHECK FOR LAST BANK FLAG.
3945 015612 001004 BNE 3$ ;BRANCH IF LAST BANK FLAG.
3946 015614 036767 163724 163760 BIT BITPT+2, LADMAP+2 ;CHECK IF LAST BANK FLAG.
3947 015622 001405 BEQ 4$ ;BR IF NOT LAST BANK.
3948 015624 016705 163746 3$: MOV LADMSK, R5 ;RESET MASK.
3949 015630 042767 020000 163736 BIC #20000, TEMPLAD ;MAKE SURE VIRTUAL LAST ADR IN BANK 2
3950 015636 016767 163700 163702 4$: MOV BITPT, TMPPT ;COPY BITPT...LO 64K.
3951 015644 016767 163674 163676 MOV BITPT+2, TMPPT+2 ;...HI 64K.
3952 015652 032705 020000 BIT #BIT13, R5 ;CHECK FOR A BLOCK SIZE OF 8K.
3953 015656 001530 BEQ 31$ ;BRANCH IF NOT.
3954 015660 013737 172344 172346 MOV 2#KIPAR2, 2#KIPAR3 ;COPY CURRENT PAR INTO FORTH PAR.
3955 015666 062737 000200 172346 5$: ADD #200, 2#KIPAR3 ;UP DATE FORTH PAR.
3956 015674 006367 163646 ASL TMPPT ;UPDATE LO POINTER TO NEXT 4K BANK.
3957 015700 006167 163644 ROL TMPPT+2 ;...HI POINTER.
3958 015704 100513 BMI 30$ ;BR IF NO MORE.
3959 015706 036767 163634 163612 6$: BIT TMPPT, TSTMAP ;CHECK IF BANK TO BE TESTED.
3960 015714 001004 BNE 7$ ;BRANCH IF A MATCH.

```



```

3961 015716 036767 163626 163604 BIT TMPPT+2,TSTMAP+2 ;CHECK FOR HI 64K BANKS.
3962 015724 001760 BEQ 5$ ;BRANCH IF NO MEMORY
3963 015726 036767 163614 163644 7$: BIT TMPPT,LADMAP ;CHECK FOR LAST BANK FLAG.
3964 015734 001004 BNE 8$ ;BRANCH IF A MATCH
3965 015736 036767 163606 163636 BIT TMPPT+2,LADMAP+2 ;CHECK HI 64K
3966 015744 001475 BEQ 31$ ;BR IF NO LAST BANK FLAG.
3967 015746 016705 163624 8$: MOV LADMSK, R5 ;RESET MASK TO FIND LAST ADDRESS.
3968 015752 052767 020000 163614 BIS #20000, TEMPLAD ;SET VIRTUAL ADR TO BANK 3.
3969 015760 000467 BR 31$
3970
3971 015762 026702 163606 10$: CMP TEMPLAD, R2 ;CHECK IF LAST ADR REACHED.
3972 015756 001064 BNE 31$ ;BR IF MORE.
3973 015770 000474 BR 32$ ;BR IF ALL DONE.
3974
3975 015772 106267 163557 20$: ASRB FLAG8K ;SHIFT 8K FLAG
3976 015776 001407 BEQ 22$ ;BR IF NOT 8K BLOCK.
3977 016000 103455 BCS 30$ ;BR IF ANOTHER 4K.
3978 016002 105067 163547 CLRB FLAG8K ;CLEAR OUT ALL FLAGS.
3979 016006 162702 040000 SUB #40000, R2 ;BACK UP 8K.
3980 016012 062702 020000 21$: ADD #20000, R2 ;UPDATE PHYSICAL ADR PNTR TO NEXT BANK.
3981 016016 106367 163520 22$: ASLB BITPT ;UPDATE POINTER.
3982 016022 100457 BMI 32$ ;BRANCH WHEN END IS REACHED.
3983 016024 036767 163512 163474 BIT BITPT, TSTMAP ;CHECK IF THIS BANK EXISTS.
3984 016032 001767 BEQ 21$ ;BRANCH IF NO MATCH.
3985 016034 036767 163502 163536 BIT BITPT, LADMAP ;CHECK FOR LAST BANK FLAG.
3986 016042 001402 BEQ 23$ ;BR IF NO MATCH.
3987 016044 016705 163526 MOV LADMSK, R5 ;RESET MASK TO FIND LAST ADR.
3988 016050 016767 163466 163470 23$: MOV BITPT, TMPPT ;SET UP TMP POINTER.
3989 016056 032705 020000 BIT #BIT13, R5 ;CHECK FOR 8K BLOCK SIZE.
3990 016062 001426 BEQ 31$ ;BRANCH IF SMALLER BLOCK SIZE.
3991 016064 106367 163456 ASLB TMPPT ;POINT TO NEXT BANK.
3992 016070 100421 BMI 30$ ;BRANCH IF OVERFLOW.
3993 016072 036767 163450 163426 BIT TMPPT, TSTMAP ;CHECK IF BANK TO BE TESTED.
3994 016100 001415 BEQ 30$ ;BRANCH IF NOT TO BE TESTED.
3995 016102 036767 163434 163470 BIT BITPT, LADMAP ;CHECK FOR LAST BANK FLAG.
3996 016110 112767 000011 163437 MOVB #11, FLAG8K ;SET 8K BLOCK FLAG.
3997 016116 036767 163420 163454 BIT BITPT, LADMAP ;CHECK FOR LAST BANK FLAG.
3998 016124 001403 BEQ 30$ ;BR IF NO FLAG.
3999 016126 016705 163444 MOV LADMSK, R5 ;RESET MASK TO FIND LAST ADR.
4000 016132 000402 BR 31$
4001 016134 012705 017777 30$: MOV #MASK4K, R5 ;SET MASK TO 4K.
4002 016140 056767 163376 163400 31$: BIS BITPT, TMPPT ;SET TMPPT FOR FINDING LAST ADR.
4003 016146 056767 163372 163374 BIS BITPT+2, TMPPT+2
4004 016154 016716 163372 MOV MMORE, (SP) ;FUDGE RETURN ADDRESS TO LOOP.
4005 016160 000207 RTS PC ;RETURN
4006
4007 016162 005767 164064 32$: TST MPRX ;* BEFORE FINAL EXIT, CHECK FOR ANY NON-TRAP PARITY ERRORS.
4008 016166 001402 BEQ 33$ ;CHECK FOR ANY PARITY REGISTERS PRESENT.
4009 016170 004767 001660 JSR PC, CKPMER ;BR IF NONE.
4010 016174 000207 33$: RTS PC ;CHECK FOR PARITY MEMORY ERRORS.
4011 ;STRAIGHT RETURN.
4012
4013 ;*****
4014 ;* MEMORY DOWNWARDS ADDRESSING SUBROUTINE.
4015 ;* FINDS NEXT LOWER 4K BANK AND UPDATES POINTERS.
4016 ;* GOES TO ADDRESS IN "MMORE" IF MORE BANKS.
4016 ;* DOES STRAIGHT EXIT WHEN ALL MEMORY HAS BEEN DONE.

```

```

4017          :*****
4018 016176 036767 163340 163362 MMDOWN: BIT BITPT, FADMAP ;CHECK FOR FIRST ADR FLAG.
4019 016204 001004          BNE 1$ ;BR IF FIRST ADR IN THIS BANK.
4020 016206 036767 163332 163354 BIT BITPT+2,FADMAP+2 ;CHECK FOR FIRST ADR FLAG.
4021 016214 001404          BEQ 2$ ;BR IF NO FLAG
4022 016216 026702 163340 1$: CMP TMPFAD, R2 ;CHECK IF FIRST ADDRESS REACHED.
4023 016222 001052          SNE 9$ ;BR IF MORE.
4024 016224 000453          BR 10$ ;BR IF ALL DONE.
4025 016226 005767 162354 2$: TST MMAPVA ;CHECK IF MEM MGMT IS AVAILABLE
4026 016232 001425          BEQ 6$ ;BRANCH IF NOT
4027 016234 162737 000200 172344 3$: SUB #200, 2*KIPAR2 ;LOWER MEM MGMT PAR BY 4K
4028 016242 006067 163276 ROR BITPT+2 ;MOV POINTER TO NEXT LOWER BANK...HI MAP.
4029 016246 006067 163270 ROR BITPT ;...LO MAP.
4030 016252 103440          BCS 10$ ;BR IF NO MORE.
4031 016254 036767 163262 163244 BIT BITPT, TSTMAP ;CHECK FOR BANK EXISTING
4032 016262 001004          BNE 4$ ;BR IF BANK TO BE TESTED.
4033 016264 036767 163254 163236 BIT BITPT+2,TSTMAP+2 ;CHECK FOR BANK IN HI MAP.
4034 016272 001760          BEQ 3$ ;BR IF NOT THERE.
4035 016274 012702 060000 4$: MOV #60000, R2 ;SET ADR POINTER TO TOP OF BANK
4036 016300 000411          BR 7$ ;GO TO COMMON EXIT
4037 016302 162702 020000 5$: SUB #20000, R2 ;BACK POINTER DOWN ONE BANK
4038 016306 006267 163230 6$: ASR BITPT ;MOVE POINTER TO NEXT LOWER BANK
4039 016312 103420          BCS 10$ ;BRANCH TO EXIT IF NO MORE MEM
4040 016314 036767 163222 163204 BIT BITPT, TSTMAP ;CHECK IF BANK EXISTS
4041 016322 001767          BEQ 5$ ;BRANCH IF BANK DOESN'T EXIST
4042 016324 036767 163212 163234 7$: BIT BITPT, FADMAP ;CHECK IF FIRST BANK FLAG.
4043 016332 001004          BNE 8$ ;BR IF FIRST BANK.
4044 016334 036767 163204 163226 BIT BITPT+2,FADMAP+2 ;CHECK IF FIRST BANK FLAG.
4045 016342 001402          BEQ 9$ ;BR IF NO FLAG FOUND.
4046 016344 016705 163214 8$: MOV FADMSK, R5 ;SET UP R5 TO FIND FIRST ADDRESS.
4047 016350 016716 163176 9$: MOV MMORE, (SP) ;RESET RETURN ADDRESS
4048 016354 000207 10$: RTS PC ;RETURN

```

M07

```

4049          .SBTTL  SUBROUTINES FOR ADDRESS AND WORSE CASE NOISE TESTS.
4050          ;*****
4051          ;*  SUBROUTINE TO CALCULATE PHYSICAL ADDRESS AND PUT IT IN R0 (BOTTOM 16 BITS).
4052          ;*  BITS 16 AND 17 ARE IN $TMP0.
4053          ;*****
4054          PHYADR: MOV     R2,    R0          ;VITRUAL INTO R0
4055          CLR     $TMP0          ;CLEAR TEMP SAVE OF HIGH BITS
4056          TST    MMAPA          ;CHECK FOR MEM MGMT AVAILABLE
4057          BEQ    1$             ;BRANCH IF NO MEM MGMT
4058          MOV    R1, -(SP)        ;:PUSH R1 ON STACK
4059          MOV    2($KIPAR2, R1    ;GET PAR TO BE ADDED TO VIRTUAL
4060          ASL    R1              ;SHIFT IT 6 TIMES
4061          ASL    R1
4062          ASL    R1
4063          ASL    R1
4064          ASL    R1
4065          ROL    $TMP0          ;SAVE EXTRA BITS
4066          ASL    R1
4067          ROL    $TMP0
4068          ADD    R1,    R0        ;ADD SHIFTED PAR TO VIRTUAL
4069          MOV    (SP)+, R1       ;:POP STACK INTO R1
4070          1$:  RTS     PC         ;RETURN
4071
4072          ;*****
4073          ;*  SUBROUTINE TO PUT BANK NUMBER INTO R0.
4074          ;*****
4075          BANKNO: CLR     R0          ;INIT R0
4076          MOV    R1, -(SP)        ;:PUSH R1 ON STACK
4077          MOV    R2, -(SP)        ;:PUSH R2 ON STACK
4078          MOV    BITPT, R1       ;GET BANK MAP POINTER...LO 64K.
4079          MOV    BITPT+2, R2     ;...HI 64K.
4080          1$:  ASR    R2          ;SHIFT POINTER...HI
4081          ROR    R1              ;...LO
4082          BCS    2$             ;BR WHEN POINTER FOUND.
4083          INCB  R0              ;COUNT BANKS.
4084          BPL    1$             ;BR IF NOT OVERFLOW.
4085          HALT                    ;FATAL ERROR!!! NO POINTER FOUND.
4086          2$:  MOV    (SP)+, R2   ;:POP STACK INTO R2
4087          MOV    (SP)+, R1       ;:POP STACK INTO R1
4088          RTS     PC         ;RETURN
4089
4090          ;*****
4091          ;*  SUBROUTINE TO WRITE THE CONSTANT IN R0 INTO ALL OF MEMORY.
4092          ;*****
4093          SETCON: JSR     R4,    INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
4094          2$:  MOV    R0,    (R2)+ ;MOV CONSTANT INTO MEMORY
4095          BIT    R5,    R2        ;CHECK FOR END OF A BLOCK.
4096          BNE   2$             ;BRANCH IF MORE IN CURRENT BLOCK.
4097          JSR   PC,    MMUP      ;FIND NEXT BLOCK AND LOOP TO 1$.
4098          RTS     PC         ;RETURN
4099
4100

```

4101
4102
4103
4104 016512 106112
4105 016514 106112
4106 016516 106112
4107 016520 106112
4108 016522 106112
4109 016524 106112
4110 016526 106112
4111 016530 106112
4112 016532 106122
4113 016534 106112
4114 016536 106112
4115 016540 106112
4116 016542 106112
4117 016544 106112
4118 016546 106112
4119 016550 106112
4120 016552 106112
4121 016554 106122
4122 016556 000207
4123
4124
4125
4126
4127 016560 012704 000020
4128
4129 016564 010022
4130 016566 010022
4131 016570 010022
4132 016572 010022
4133
4134 016574 010322
4135 016576 010322
4136 016600 010322
4137 016602 010322
4138
4139 016604 010022
4140 016606 010022
4141 016610 010022
4142 016612 010022
4143
4144 016614 010322
4145 016616 010322
4146 016620 010322
4147 016622 010322
4148
4149 016624 005304
4150 016626 001356
4151 016630 010046
4152 016632 010300
4153 016634 012603
4154 016636 000207

```
*****  
;* 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
```

```
*****  
;* SUBROUTINE TO WRITE 3 XOR 9 PATTERN INTO 256. WORD BLOCK.  
*****  
W3X9: MOV #16.,R4 ;EACH LOOP WRITES 256. WORDS  
2$: MOV R0,(R2)+  
MOV R0,(R2)+  
MOV R0,(R2)+  
MOV R0,(R2)+  
MOV R3,(R2)+  
MOV R3,(R2)+  
MOV R3,(R2)+  
MOV R3,(R2)+  
MOV R0,(R2)+  
MOV R0,(R2)+  
MOV R0,(R2)+  
MOV R0,(R2)+  
MOV R3,(R2)+  
MOV R3,(R2)+  
MOV R3,(R2)+  
MOV R3,(R2)+  
DEC R4  
BNE 2$  
MOV R0,-(SP) ;SAVE R0  
MOV R3,R0 ;PUT R3 INTO R0  
MOV (SP)+,R3 ;PUT SAVED R0 INTO R3  
RTS PC ;RETURN
```

```

4155
4156
4157
4158
4159 016640
4160 016640 010246
4161 016642 010346
4162 016644 010446
4163 016646 012502
4164 016650 012503
4165 016652 012704 020000
4166 016656 012223
4167 016650 005304
4168 016662 001375
4169 016664 012704 020000
4170 016670 024243
4171 016672 001417
4172 016674 011267 162224
4173 016700 011367 162222
4174 016704 010267 162210
4175 016710 010367 162206
4176 016714 004767 003116
4177 016720 000023
4178 016722 000000
4179 016724 162705 000004
4180 016730 000746
4181 016732 005304
4182 016734 001355
4183 016736 004567 004275
4184 016742 026121
4185
4186 016744 010346
4187 016746 004767 005722
4188 016752 012604
4189 016754 012603
4190 016756 012602
4191 016760 000205
4192
4193 016762 022767 000033 161612
4194 016770 001401
4195 016772 000000
4196 016774
4197 016774 010046
4198 016776 010146
4200 017000 005767 161602
4201 017004 001465
4202 017006 012737 007600 172346
4203 017014 005000
4204 017016 012701 100000
4205 017022 162737 000200 172346
4206 017030 006001
4207 017032 006000
4208 017034 103500
4209 017036 030167 162462

```

```

.SBTTL RELOCATION SUBROUTINES.
*****
* ROUTINE TO RELOCATE PROGRAM CODE
*****
RELOC:
      MOV     R2, -(SP)      ;; PUSH R2 ON STACK
      MOV     R3, -(SP)      ;; PUSH R3 ON STACK
      MOV     R4, -(SP)      ;; PUSH R4 ON STACK
4S:   MOV     (R5)+, R2      ;; GET FIRST LOCATION.
      MOV     (R5)+, R3      ;; GET FIRST LOCATION OF DESTINATION.
      MOV     #20000, R4     ;; SET UP BK COUNTER.
1S:   MOV     (R2)+, (R3)+  ;; MOV THE DATA.
      DEC     R4             ;; COUNT THE WORDS.
      BNE    #S             ;; BR IF MORE.
      MOV     #20000, R4     ;; RESET THE COUNTER.
2S:   CMP     -(R2), -(R3)  ;; CHECK THE DATA JUST MOVED.
      BEQ    #S             ;; BR IF DATA OK.
      MOV     (R2), $GDADR  ;; GET SOURCE DATA.
      MOV     (R3), $DDADR  ;; GET DESTINATION DATA.
      MOV     R2, $GDADR    ;; GET SOURCE ADDRESS.
      MOV     R3, $DDADR    ;; GET DESTINATION ADDRESS.
      JSR    PC, $ERRR      ;; ** ERROR ** (GO TYPE A MESSAGE)
      .WORD  R3             ;; ERROR TYPE CODE.
      HALT   $FATAL        ;; FATAL ERROR!!! RELOCATION FAILED.
      SUB    #4, R5         ;; ADJUST RETURN POINTER.
      BR    4S             ;; GO BACK AND TRY AGAIN.
3S:   DEC     R4             ;; COUNT WORDS.
      BNE    #S             ;; BR IF MORE.
      JSR    R5, $PRINT     ;; GO PRINT OUT THE FOLLOWING MESSAGE.
      .WORD  $PRELOC        ;; ADDRESS OF MESSAGE TO BE TYPED
      MOV     R3, -(SP)     ;; PUT THE DATA ON THE STACK.
      JSR    PC, $STYPAC    ;; DETERMINE THE PHYSICAL ADDRESS AND TYPE IT.
      MOV     (SP)+, R4     ;; POP STACK INTO R4
      MOV     (SP)+, R3     ;; POP STACK INTO R3
      MOV     (SP)+, R2     ;; POP STACK INTO R2
      RTS    R5             ;; RETURN
*****
* SUBROUTINE TO MOVE PROGRAM FROM BOTTOM TO TOP OF MEMORY.
*****
RELTOP: CMP     #3, PRGMAP  ;; CHECK THAT THE PROGRAM IS NOW IN BANKS 0 AND 1.
      BEQ    #S             ;; BR IF OK
      HALT   $FATAL        ;; FATAL ERROR!!! PROG SHOULD BE IN BANKS 0 AND 1
1S:   MOV     R0, -(SP)     ;; PUSH R0 ON STACK
      MOV     R1, -(SP)     ;; PUSH R1 ON STACK
      TST    MMABA
      BEQ    #S
      MOV     #7600, @#KIPAR3 ;; SET PAR TO TOP OF MEM
      CLR    R0             ;; INIT BANK POINTER...LO 64K
      MOV     @BIT15, R1    ;; ...HI 64K.
2S:   SUB    #200, @#KIPAR3 ;; BACK DOWN ONE BANK.
      ROR    R1             ;; MOVE POINTER...HI 64K.
      ROR    R0             ;; ...LO 64K.
      BCS    #S
      BIT    R1, MEMMAP+2  ;; CHECK FOR BANK EXISTS.

```

```

4211 017042 001003 BNE 3$ :BR IF AVAILABLE
4212 017044 030067 162452 BIT RO, MEMMAP :CHECK FOR BANK EXISTS.
4213 017050 001764 BEQ 2$ :BR IF NO BANK FOUND.
4214 017052 013737 172346 172344 3$: MOV J#KIPAR3,J#KIPAR2 :COPY PAR
4215 017060 010046 MOV RO,-(SP) :PUSH RO ON STACK
4216 017062 010146 MOV RI,-(SP) :PUSH RI ON STACK
4217 017064 162737 000200 172344 4$: SUB #200, J#KIPAR2 :BACK DOWN WITH LOW PAR.
4218 017072 00E201 ROR RI :SHIFT POINTER.
4219 017074 006000 ROR RO :...LO 64K.
4220 017076 103457 BCS 90$ :BR IF OVERFLOW.
4221 017100 030167 162420 5$: BIT RI, MEMMAP+2 :CHECK IF BANK EXISTS...HI 64K.
4222 017104 001003 BNE 6$ :BR IF BANK EXISTS.
4223 017106 030067 162410 BIT RO, MEMMAP :CHECK IF BANK EXISTS...LO 64K.
4224 017112 001764 BEQ 4$ :BR IF BANK DOESN'T EXIST.
4225 017114 052601 6$: BIS (SP)+, RI :GET SECOND BANK POINTER.
4226 017116 052600 BIS (SP)+, RO :...LO 64K.
4227 017120 030067 161456 BIT RO, PRGMAP :CHECK FOR CONFLICT.
4228 017124 001044 BNE 90$ :ABORT IF DESTINATION OVERLAYS SOURCE.
4229 017126 004567 177506 JSR R5, RELOC :GO RELOCATE PROGRAM.
4230 017132 000000 .WORD 0 :SOURCE FIRST ADDRESS.
4231 017134 040000 .WORD 40000 :DESTINATION FIRST ADDRESS.
4232 017136 013737 172344 172343 MOV J#KIPAR2,J#KIPAR0 :RELOCATE LO BANK
4233 017144 013737 172346 172342 MOV J#KIPAR3,J#KIPAR1 :RELOCATE HI BANK.
4234 :* PROGRAM SHOULD NOW BE EXECUTING OUT OF LAST TWO BANKS OF MEMORY.
4235 017152 010167 161426 MOV RI, PRGMAP+2 :RESET PROGRAM MAP.
4236 017156 000473 BR 30$ :BR TO COMMON EXIT.
4238 017160 012700 000400 10$: MOV #BIT8, RO :SET BANK POINTER TO TOP OF MEM.
4239 017164 00E201 CLR RI :SET ADDRESS POINTER TO TOP.
4240 017166 162701 020000 11$: SUB #20000, RI :BACK DOWN ONE BANK.
4241 017172 006200 ASR RO :MOVE POINTER DOWN ONE BANK.
4242 017174 103420 BCS 90$ :BR IF OVERFLOW.
4243 017176 030067 162320 BIT RO, MEMMAP :CHECK IF THIS BANK EXISTS.
4244 017202 001771 BEQ 11$ :BR IF NON-EXISTANT BANK.
4245 017204 162701 020000 SUB #20000, RI :BACK DOWN TO NEXT BANK.
4246 017210 005200 ASR RO :MOV POINTER DOWN ONE BANK.
4247 017212 103411 BCS 90$ :BR IF OVERFLOW.
4248 017214 030067 162302 BIT RO, MEMMAP :CHECK IF THIS BANK EXISTS.
4249 017220 001762 BEQ 11$ :BR TO START OVER IF NO LOWER BANK.
4250 017222 010046 MOV RO,-(SP) :SAVE THE POINTER.
4251 017224 006300 ASL RO :RESET POINTER TO HI BANK.
4252 017226 052600 BIS (SP)+, RO :SET BIT FOR LO BANK.
4253 017230 030067 161346 BIT RO, PRGMAP :CHECK FOR A PROGRAM CONFLICT.
4254 017234 001401 BEQ 12$ :BR IF NO CONFLICT.
4255 017236 90$: HALT :FATAL ERROR!!! NOT ENOUGH MEMORY??
4256 017236 000000 12$: MOV RI, 13$ :SET DATA FOR RELOCATION SUBROUTINE.
4257 017240 010167 000006 JSR R5, RELOC :GO RELOCATE THE PROGRAM TO TOP OF MEM.
4258 017244 004567 177370 .WORD 0 :SOURCE STARTING ADDRESS.
4259 017250 000000 .WORD 0 :DESTINATION STARTING ADDRESS.
4260 017252 000000 13$: .WORD 0 :DESTINATION STARTING ADDRESS.
4261 017254 010167 161320 MOV RI, RELOCF :SET RELOCATION FACTOR IN UNRELOCATED CODE.
4262 017260 060107 ADD RI, PC :JUMP TO RELOCATED PROGRAM
4263 :* PROGRAM NOW EXECUTING OUT OF TOP OF MEMORY.
4264 017262 060106 ADD RI, SP :ADJUST THE STACK POINTER TO TOP OF MEMORY.
4265 017264 010167 161310 MOV RI, RELOCF :SET THE RELOCATION FACTOR.
4266 017270 060137 000004 ADD RI, J#ERRVEC :ADJUST ERROR VECTOR.

```

```

4286 017274 060137 000024      ADD      R1,      @#PWRVEC ;ADJUST POWER FAIL VECTOR.
4287 017300 060137 000114      ADD      R1,      @#PARVEC ;ADJUST PARITY ERROR VECTOR.
4288 017304 026727 161626 177570    CMP      SWR,      @177570 ;CHECK FOR HARDWARE SWITCH REGISTER.
4289 017312 001404          BEQ      14$      ;BR IF HARDWARE SWITCH REGISTER.
4290 017314 060167 161616      ADD      R1,      SWR      ;ADJUST SOFTWARE SWITCH REGISTER.
4291 017320 060167 161614      ADD      R1,      DISPLAY ;ADJUST SOFTWARE DISPLAY REGISTER.
4292 017324 062701 001610 14$:      ADD      @RADTAB,R1 ;POINT TO THE RELATIVE RELOCATION TABLE.
4293 017330 066721 161244 15$:      ADD      RELOCF, (R1)+ ;ADD RELOCATION FACTOR TO ADDRESSES IN TABLE.
4294 017334 005721 16$      TST      (R1)+      ;CHECK FOR INTERUM TERMINATOR.
4295 017336 001776          BEQ      16$      ;BR SO AS TO NOT MODIFY ZERO.
4296 017340 024127 177777    CMP      -(R1),    #-1    ;CHECK FOR END OF TABLE.
4297 017344 001371          BNE      15$      ;BR IF MORE IN TABLE.
4298 017346 010067 161230 30$:      MOV      RO,      PRGMAP ;SET NEW PROGRAM MAP...LO 64K.
4299 017352 012601          MOV      (SP)+,R1 ;POP STACK INTO R1
4300 017354 012600          MOV      (SP)+,RO ;POP STACK INTO RO
4301 017356 066716 161216      ADD      RELOCF, (SP) ;ADJUST RETURN ADDRESS.
4302 017362 000207          RTS      PC      ;RETURN

:*****
:* SUBROUTINE TO RELOCATE PROGRAM BACK TO BANKS 0 AND 1.
:*****
4288 017364 032767 000003 161210 RELO:   BIT      #3,      PRGMAP ;CHECK FOR PROGRAM ALREADY IN BANKS 0 OR 1.
4289 017372 001401          BEQ      1$      ;BR IF NO CONFLICT.
4290 017374 000000          HALT          ;FATAL ERROR!!! PROGRAM ALREADY IN BANKS 0 OR 1!!!!
4291 017376 005767 161204 1$:      TST      MMABA      ;CHECK FOR MEM MGMT.
4292 017402 001417          BEQ      10$     ;BR IF NO MEMMGMT.
4293 017404 005037 172344      CLR      @#KIPAR2 ;SET PAR 2 TO BANK 0.
4294 017410 012737 000200 172346      MOV      #200,    @#KIPAR3 ;SET PAR 3 TO BANK 1.
4295 017416 004567 177216      JSR      R5,      RELOC ;GO MOVE BK INTO BANKS 0 AND 1.
4296 017422 000000          .WORD      0      ;SOURCE STARTING ADDRESS.
4297 017424 040000          .WORD      40000   ;DESTINATION STARTING ADDRESS.
4298 017426 005037 172340      CLR      @#KIPAR0 ;RESTORE PAR 0 TO BANK 0.
4299 017432 012737 000200 172342      MOV      #200,    @#KIPAR1 ;RESTORE PAR 1 TO BANK 1.
4300          :* PROGRAM IS NOW EXECUTING OUT OF BANKS 0 AND 1.
4301 017440 000444          BR      30$     ;BR TO COMMON EXIT.

4303 017442 016746 161132 10$:      MOV      RELOCF, -(SP) ;PUT RELOCATION FACTOR ONTO THE STACK.
4304 017446 011667 000004      MOV      (SP),    20$   ;SET DATA FOR RELOC SUBROUTINE.
4305 017452 004567 177152      JSR      R5,      RELOC ;GO MOVE THE PROGRAM BACK TO BANKS 0 AND 1.
4306 017456 000000          .WORD      0      ;SOURCE STARTING ADDRESS.
4307 017460 000000          .WORD      0      ;DESTINATION STARTING ADDRESS.
4308 017462 161607          SUB      (SP),    PC ;JUMP TO RELOCATED PROGRAM.
4309          :* THE PROGRAM IS NOW EXECUTING OUT OF BANKS 0 AND 1.
4310          SUB      (SP),    SP ;RESET THE STACK POINTER.
4311          MOV      RO,    -(SP) ;PUSH RO ON STACK
4312 017470 012700 001610      MOV      @RADTAB,RO ;SET UP POINTER TO RELATIVE ADDRESS TABLE.
4313 017474 166620 000002 21$:      SUB      2(SP),   (RO)+ ;RESET ADDRESSES TO UNRELOCATED VALUES.
4314 017500 005720 22$:      TST      (RO)+      ;CHECK FOR TERMINATORS.
4315 017502 001776          BEQ      22$      ;BR OVER TERMINATORS.
4316 017504 024027 177777    CMP      -(RO),    #-1    ;CHECK FOR END OF TABLE INDICATOR.
4317 017510 001371          BNE      21$      ;BR IF MORE ADDRESSES IN TABLE.
4318 017512 012600          MOV      (SP)+,RO ;POP STACK INTO RO
4319 017514 161637 000004      SUB      (SP),    @#ERRVEC ;ADJUST ERROR VECTOR.
4320 017520 161637 000024      SUB      (SP),    @#PWRVEC ;ADJUST POWER FAIL VECTOR.
4321 017524 161637 000114      SUB      (SP),    @#PARVEC ;ADJUST PARITY ERROR VECTOR.
4322 017530 026727 161402 177570    CMP      SWR,      @177570 ;CHECK FOR HARDWARE SWITCH REGISTER.

```

E08

4333	017536	001404		BEQ	23\$:BR IF HARDWARE SWITCH REGISTER.
4334	017540	161667	161372	SUB	(SP),	SWR	:ADJUST SOFTWARE SWITCH REGISTER.
4335	017544	161667	161370	SUB	(SP),	DISPLAY	:ADJUST SOFTWARE DISPLAY REGISTER.
4336	017550	162616		SUB	(SP)+	(SP)	:ADJUST RETURN ADDRESS.
4337	017552	005067	161022	CLR	RELOC#		:RESET RELOCATION FACTOR.
4338	017556	012767	000003	MOV	#3	PRGMAP	:SET PROGRAM MAP TO POINT TO BANKS 0 AND 1.
4339	017564	005067	161014	CLR	PRGMAP+2		:...HI 64K.
4330	017570	000207		RTS	PC		:RETURN.

161016

23\$:
30\$:

 : * THIS SUBROUTINE MOVES THE LOADER AREA BACK TO THE "TOP" OF MEMORY FROM
 : * WHENCE IT CAME. THE LOADER AREA IS SAVED AT THE END OF THE BK OF
 : * PROGRAM CODE WHEN THE PROGRAM IS INITIALLY RUN.

4337	017572	016700	161720	RESLDR: MOV	LMAD,	RC	:CHECK IF THE LOADERS WERE SAVED.
4338	017576	001001		BNE	IS		:BR IF LOADER AREA WAS SAVED.
4339	017600	000000		HALT			:FATAL ERROR!!! CAN'T RESTORE LOADER AREA IF IT WASN'T SAVED.
4340	017602	005767	161000	1\$: TST	MMAVA		:CHECK FOR MEM MGMT.
4341	017606	001402		BEQ	2\$:SKIP IF NO MEM MGMT.
4342	017610	005037	177572	CLR	2\$SRC		:DISABLE MEM MGMT.
4343	017614	012701	040000	2\$: MOV	#4000,	R1	:GET END OF BK, ASSUME PROG NOT RELOCATED.
4344	017620	012702	002734	MOV	#1500,	R2	:GET COUNTER.
4345	017624	014140		3\$: MOV	-(R1),	-(R0)	:MOVE THE LOADER AREA.
4346	017626	005302		DEC	R2		:COUNT HOW LONG THE AREA IS.
4347	017630	001375		BNE	3\$:BR IF NOT MORE TO MOVE.
4348	017632	005767	160750	TST	MMAVA		:CHECK FOR MEM MGMT.
4349	017636	001402		BEQ	4\$:BR IF NO MEM MGMT.
4350	017640	005237	177572	INC	2\$SRC		:ENABLE MEM MGMT.
4351	017644	000207		4\$: RTS	PC		:RETURN.


```

4352 .SBTTL PARITY MEMORY TRAP SERVICE AND SUBROUTINES.
4353 ;*****
4354 ;* PARITY MEMORY UNEXPECTED ERROR TRAP SERVICE ROUTINE.
4355 ;* FIND OUT WHICH REGISTER DETECTED THE ERROR.
4356 ;* THEN SCAN MEMORY TO SEE IF PARITY ERROR STILL SET AND REPORT LOCATION.
4357 ;*****
4358 017646 011667 161250 PESRV: MOV (SP), SBDADR ;GET PC OF INSTRUCTION WHICH CAUSED ERROR.
4359 017652 004567 003362 JSR R5, SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
4360 017655 26060 .WORD UNEXPT ;ADDRESS OF MESSAGE TO BE TYPED
4361 ;"UNEXPECTED MEMORY PARITY TRAP."
4362 017660 010146 MOV R1, -(SP) ;PUSH R1 ON STACK
4363 017662 010346 MOV R3, -(SP) ;PUSH R3 ON STACK
4364 017664 016703 161726 MOV MPRX, R3 ;GET POINTER TO PARITY REGISTERS.
4365 017670 005733 1$: TST @R3+ ;CHECK THE PARITY REG FOR AN ERROR FLAG.
4366 017672 100415 BMI 3$ ;BR IF THIS REGISTER SHOWS THE ERROR.
4367 017674 005713 TST R3 ;CHECK FOR TABLE TERMINATOR.
4368 017676 001374 BNE 1$ ;BR IF MORE REGISTERS.
4369 017700 004767 002132 JSR PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)
4370 ;***ERROR*** NO REGISTER INDICATED ERROR
4371 017704 000024 .WORD 24 ;ERROR TYPE CODE.
4372 017706 000417 BR 4$ ;EXIT
4373 017710 005713 2$: TST R3 ;CHECK FOR TABLE TERMINATOR.
4374 017712 001415 BEQ 4$ ;BR IF NO MORE PARITY REGISTERS.
4375 017714 005733 TST @R3+ ;CHECK THE PARITY REG FOR AN ERROR FLAG.
4376 017716 100374 BPL 2$ ;BR IF NO ERROR FLAG.
4377 017720 004567 003314 JSR R5, SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
4378 017724 026151 .WORD MTOE ;ADDRESS OF MESSAGE TO BE TYPED
4379 ;"MORE THAN ONE ERROR FOUND."
4380 017726 3$: JSR PC, SPRINTQ ;SET UP VALUES FOR ERROR PRINTING.
4381 017726 004767 000610 64$: JSR PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)
4382 017732 004767 002100 .WORD 25 ;ERROR TYPE CODE.
4383 017736 000025 JSR PC, PSCAN ;GO SCAN MEMORY FOR BAD PARITY.
4384 017740 004767 000216 BR 2$ ;GO LOOK FOR MORE ERRORS.
4385 017744 000761 4$: MOV (SP)+, R3 ;POP STACK INTO R3
4386 017746 012603 MOV (SP)+, R1 ;POP STACK INTO R1
4387 017750 012601 RTI ;RETURN.
4388 017752 000002
4390 ;*****
4391 ;ROUTINE TO ENABLE PARITY ERROR ACTION ON MA/MF PARITY MEMORIES
4392 ;THIS ROUTINE IS MEANT TO CATCH UNEXPECTEDS
4393 ;*****
4394 MAMF: TST MPRX ;CHECK IF ANY PARITY REGISTERS EXIST.
4395 017754 005767 162272 BEQ MAMF2 ;EXIT IF NO PARITY REGISTERS.
4396 017760 001434 BIT #SW6, @SWR ;CHECK FOR INHIBIT PARITY ERROR DETECTION.
4397 017762 032777 000100 161146 BNE MAMF2 ;EXIT IF NO PARITY ERROR DETECTION.
4398 017770 001030 TST RELOCF ;CHECK IF PROGRAM RELOCATED OUT OF BANK J.
4399 017772 005767 160602 BEQ SETAE ;BR IF PROG IN BANK 0.
4400 017776 001410 BIT #SW5, @SWR ;CHECK IF VECTORS PROTECTED.
4401 020000 032777 000040 161130 BNE SETAE ;BR IF VECTOR AREA PROTECTED.
4402 020006 001004 CMP FSTADR, #1000 ;CHECK FOR STARTING ADDRESS ABOVE THE VECTORS.
4403 020010 026727 161544 001000 BLO MAMF2 ;EXIT IF VECTORS EXPOSED TO TESTING.
4404 020016 103415 SETAE: MOV PESRV, @PARVEC ;SET PARITY ERROR TRAP VECTOR
4405 020020 016737 161606 000114 CLR @PARVEC+2 ;PRIORITY LEVEL 0 ON TRAP
4406 020026 005037 000116 MOV R3, -(SP) ;PUSH R3 ON STACK
4407 020032 010346

```

4408 020034 016703 161556
4409 020040 052733 000001
4410 020044 005713
4411 020046 001374
4412 020050 012603
4413 020052 000207
4414
4415
4416
4417
4418 020054 005767 162172
4419 020050 001437
4420 020052 032777 000100 161046
4421 020070 001033
4422 020072 010346
4423 020074 016703 161516
4424 020100 005733 15:
4425 020102 100023
4426 020104 032773 000001 177776
4427 020112 001010
4428 020114 004767 000422 645:
4429 020120 004767 001712
4430 020124 000026
4431 020126 000411
4432 020130 004767 000026
4433 020134
4434 020134 004767 000402 25:
4435 020140 004767 001672 655:
4436 020144 000027
4437 020146 004767 000010
4438 020152 005713 35:
4439 020154 001351
4440 020156 012603
4441 020160 000207 45:
4442
4443
4444
4445
4446
4447
4448 020162
4449 020162 010046
4450 020164 010146
4451 020166 010246
4452 020170 010346
4453 020172 010446
4454 020174 013746 000114
4455 020200 013746 000116
4456 020204 004567 003030
4457 020210 026215
4458
4459 020212 012700 000001
4460 020216 005001
4461 020220 005002
4462 020222 005004
4463 020224 004767 000256

```
MAMF1: MOV .MPRX, R3 ;GET PARITY REGISTER TABLE POINTER.  
;SET ACTION ENABLE BIT IN PARITY REG  
BIS #AE, 2(R3)+  
TST (R3) ;CHECK FOR END OF TABLE.  
BNE MAMF1 ;BR IF MORE PARITY REGISTERS.  
MOV (SP)+,R3 ;POP STACK INTO R3  
MAMF2: RTS PC ;RETURN.
```

* SUBROUTINE TO CHECK PARITY REGISTERS FOR ERRORS THAT DIDN'T TRAP.

```
CKPMER: TST MPRX ;CHECK IF ANY PARITY REGISTERS EXIST.  
BEQ 45 ;BR IF NO PARITY REGISTERS.  
BIT #SW6, 2SWR ;CHECK FOR INHIBIT PARITY ERROR CHECKING.  
BNE 45 ;BR IF PARITY ERROR CHECKING INHIBITED.  
MOV R3, -(SP) ;PUSH R3 ON STACK  
MOV .MPRX, R3 ;GET PARITY REG TABLE POINTER.  
15: TST 2(R3)+ ;CHECK THE PARITY REG FOR AN ERROR FLAG.  
BPL 35 ;BR IF NO ERROR  
BIT #BIT0, 2-2(R3) ;CHECK IF A TRAP SHOULD HAVE OCCURRED.  
BNE 25 ;BR IF NO ACTION ENABLE.  
645: JSR PC, SPRTQ ;SET UP VALUES FOR ERROR PRINTING.  
JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)  
.WORD 26 ;ERROR TYPE CODE.  
BR 35  
JSR PC, PSCAN ;GO SCAN ALL MEMORY FOR PARITY ERRORS.  
25:  
655: JSR PC, SPRTQ ;SET UP VALUES FOR ERROR PRINTING.  
JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)  
.WORD 27 ;ERROR TYPE CODE.  
JSR PC, PSCAN ;GO SCAN ALL MEMORY FOR PARITY ERRORS.  
35: TST (R3) ;CHECK FOR TABLE TERMINATOR.  
BNE 15 ;BR IF MORE.  
MOV (SP)+,R3 ;POP STACK INTO R3  
45: RTS PC ;RETURN.
```

* THIS SUBROUTINE WILL SCAN ALL OF MEMORY LOOKING FOR BAD PARITY.
* TYPE OUT ALL LOCATIONS FOUND TO BE BAD, AND WRITE BACK INTO THE
* LOCATIONS IN ORDER TO RESTORE GOOD PARITY.

```
PSCAN:  
MOV R0, -(SP) ;PUSH R0 ON STACK  
MOV R1, -(SP) ;PUSH R1 ON STACK  
MOV R2, -(SP) ;PUSH R2 ON STACK  
MOV R3, -(SP) ;PUSH R3 ON STACK  
MOV R4, -(SP) ;PUSH R4 ON STACK  
MOV 2#114, -(SP) ;PUSH 2#114 ON STACK  
MOV 2#116, -(SP) ;PUSH 2#116 ON STACK  
JSR R5, $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.  
.WORD SCANM ;ADDRESS OF MESSAGE TO BE TYPED  
;"SCANNING MEMORY FOR BAD PARITY."  
MOV #BIT0, R0 ;SET BIT POINTER TO FIRST BANK.  
CLR R1 ;CLR HI 64K POINTER.  
CLR R2 ;INIT ADDRESS POINTER.  
CLR R4 ;INIT ERROR DETECTED FLAG.  
JSR PC, CLRPAR ;CLEAR THE PARITY REGISTERS.
```

```

4464 020230 012737 000116 000114      MOV      #116,  @#114      ;HALT IF ANOTHER PARITY TRAP.
4465 020236 005037 000116              CLR      @#116
4466 020242 005767 160340              TST     MMAVA              ;CHECK FOR MEMORY MANAGEMENT.
4467 020246 001406              BEQ     1$                 ;BR IF NO MEM MGMT.
4468 020250 013746 172344              MOV     @#KIPAR2,-(SP)    ;PUSH @#KIPAR2 ON STACK
4469 020254 005037 172344              CLR     @#KIPAR2         ;INIT MEM MGMT TO POINT TO BANK 0.
4470 020260 012702 040000              MOV     #40000, R2       ;SET ADR POINTER TO PAR2.
4471 020264 030067 161232      1$:      BIT     R0,  MEMMAP    ;CHECK IF THIS BANK OF MEM EXISTS.
4472 020270 001003              BNE    2$                 ;BR IF THIS BANK EXISTS.
4473 020272 030167 161226              BIT     R1,  MEMMAP+2    ;CHECK HI 64K MAP.
4474 020276 001442              BEQ     10$              ;BR IF THIS BANK DOESN'T EXIST.
4475 020300      2$:
4476 020300 010146              MOV     R1,-(SP)         ;PUSH R1 ON STACK
4477 020302 111201      3$:      MOVVB  (R2),  R1          ;READ THE LOCATION TO SEE IF IT HAS A PARITY ERROR.
4478 020304 016703 161306              MOV     .MPRX,  R3       ;SET UP POINTER TO PARITY REGISTERS.
4479 020310 005733      4$:      TST     @R3)+           ;CHECK FOR THE ERROR FLAG.
4480 020312 100024              BPL    6$                 ;BR IF NO ERROR FLAG.
4481 020314 005704              TST     R4                ;CHECK IF FIRST ERROR, THIS SCAN.
4482 020316 001003              BNE    5$                 ;BR IF MORE THAN ONE ERROR FOUND.
4483 020320 005367 160566              DEC     $ERTTL          ;ADJUST ERROR COUNT.
4484 020324 005204              INC     R4                ;SET FLAG TO INDICATE ERROR FOUND.
4485 020326      5$:
4486 020326 004767 000210      6$:      JSR     PC,  SPRNTQ     ;SET UP VALUES FOR ERROR PRINTING.
4487 020332 004767 001500              JSR     PC,  $ERROR     ;*** ERROR *** (GO TYPE A MESSAGE)
4488 020336 000030              .WORD  30                ;ERROR TYPE CODE.
4489 020340 111212              MOVVB  (R2),  (R2)       ;REWRITE THE LOCATION TO CLEAR BAD PARITY.
4490 020342 005053              CLR     @-(R3)          ;CLEAR THE ERROR FLAG.
4491 020344 105712              TSTB   (R2)             ;CHECK IF THE PARITY ERROR WAS CLEARED.
4492 020346 005733              TST     @R3)+           ;CHECK FOR THE ERROR FLAG.
4493 020350 100005              BPL    6$                 ;BR IF IT IS OK.
4494 020352 004567 002662              JSR     R5,  $PRINT     ;GO PRINT OUT THE FOLLOWING MESSAGE.
4495 020356 025257              .WORD  PEWNC            ;ADDRESS OF MESSAGE TO BE TYPED
4496                                ;"PARITY ERROR WILL NOT CLEAR."
4497 020360 005073 177775              CLR     @-2(R3)         ;CLEAR OUT THE PARITY ERROR FLAG.
4498 020364 005713      7$:      TST     (R3)            ;CHECK FOR THE END OF REG ADR TABLE.
4499 020366 001350              BNE    4$                 ;BR IF MORE PARITY REGISTERS.
4500 020370 005202              INC     R2                ;GO TO NEXT MEMORY ADDRESS.
4501 020372 032702 017777              BIT     #MASK4K,R2      ;CHECK FOR END OF 4K BANK.
4502 020376 001341              BNE    3$                 ;BR IF MORE MEMORY THIS BANK.
4503 020400 012601              MOV     (SP)+,R1        ;POP STACK INTO R1
4504 020402 000402              BR     11$              ;BR TO CHECK FOR NEXT BANK.
4505 020404 062702 020000      10$:     ADD     #20000, R2       ;SKIP BANKS THAT AREN'T THERE.
4506 020410 005767 160172      11$:     TST     MMAVA          ;CHECK FOR MEM MGMT.
4507 020414 001413              BEQ     12$              ;BR IF NO MEM MGMT.
4508 020416 062737 000200 172344      ADD     #200,  @#KIPAR2 ;UPDATE MEM MGMT REG TO NEXT 4K.
4509 020424 012702 040000              MOV     #40000, R2     ;RESET ADDRESS POINTER TO BEGINNING OF BANK.
4510 020430 006300              ASL    R0                ;UPDATE BANK POINTER.
4511 020432 006101              ROL    R1                ;...HI 64K.
4512 020434 100313              BPL    1$                 ;BR IF MORE BANKS.
4513 020436 012637 172344              MOV     (SP)+,@#KIPAR2 ;POP STACK INTO @#KIPAR2
4514 020442 000402              BR     20$              ;GO CHECK IF ANY ERRORS FOUND.
4515 020444 106300      12$:     ASLB   R0                ;UPDATE POINTER TO NEXT BANK.
4516 020446 100306              BPL    1$                 ;BR IF MORE BANKS.
4517 020450 005704      20$:     TST     R4                ;CHECK IF ANY PARITY ERRORS DETECTED.
4518 020452 001003              BNE    21$              ;BR IF ERRORS DETECTED.
4519 020454 004567 002560              JSR     R5,  $PRINT     ;GO PRINT OUT THE FOLLOWING MESSAGE.

```

```

4520 020460 025430          .WORD  NOPES          ;ADDRESS OF MESSAGE TO BE TYPED
4521 020462          21$:          MOV      (SP)+,0#116      ;;POP STACK INTO 0#116
4522 020462 012637 000116      MOV      (SP)+,0#114      ;;POP STACK INTO 0#114
4523 020466 012637 000114      MOV      (SP)+,R4        ;;POP STACK INTO R4
4524 020472 012604          MOV      (SP)+,R3        ;;POP STACK INTO R3
4525 020474 012603          MOV      (SP)+,R2        ;;POP STACK INTO R2
4526 020476 012602          MOV      (SP)+,R1        ;;POP STACK INTO R1
4527 020500 012601          MOV      (SP)+,R0        ;;POP STACK INTO R0
4528 020502 012600          MOV      (SP)+,R0        ;;POP STACK INTO R0
4529 020504 000207          RTS      PC              ;RETURN.
4530
4531 ;*****
4532 ;ROUTINE TO CLEAR ALL PARITY REGISTERS PRESENT
4533 ;*****
4534 CLRPAR:
4535 020506 010346          MOV      R3, -(SP)       ;;PUSH R3 ON STACK
4536 020510 016703 161102      MOV      .MPRX, R3      ;GET PARITY REGISTER TABLE POINTER.
4537 020514 005713          1$:      TST      (R3)         ;CHECK FOR THE TABLE TERMINATOR.
4538 020516 001402          BEQ      2$            ;BR IF DONE ALL PARITY REGISTERS.
4539 020520 005033          CLR      0(R3)+        ;CLEAR THE PARITY REGISTER.
4540 020522 000774          BR      1$            ;BR FOR MORE
4541 020524
4542 020524 012603          2$:      MOV      (SP)+,R3     ;;POP STACK INTO R3
4543 020526 000207          RTS      PC              ;RETURN.
4544
4545 .SBTTL SUBROUTINES TO SET UP DATA FOR ERROR PRINTOUT ROUTINE.
4546 ;*****
4547 ;* THESE ROUTINES ARE USED TO TRANSFER DATA TO COMMON TAG AREA (.SCMTAG)
4548 ;* FOR ERROR PRINTOUT BY .SERROR & .SERRYP ROUTINES FROM **SYSTEMAC**.
4549 ;*****
4550 020530 010267 160364      SPRNT:  MOV      R2,      $GDADR ;SAVE THE ADDRESS UNDER TEST.
4551 020534 005067 160364      CLR      $GDDAT ;SHOULD BE DATA IS "0".
4552 020540 000430          BR      SPRNTB
4553
4554 020542 014367 160410      SPRNTQ: MOV      -(R3),  $TMPD ;GET THE PARITY REGISTER ADDRESS.
4555 020546 013367 160406      MOV      0(R3)+,  $TMP1 ;GET THE CONTENTS OF THE PARITY REG.
4556 020552 000402          BR      SPRNTC
4557
4558 020554 011367 160376      SPRNTP: MOV      (R3),  $TMPD ;GET THE PARITY REGISTER ADDRESS.
4559 020560 010267 160334      SPRNTO: MOV      R2,      $GDADR ;GET THE MEMORY ADDRESS BEING TESTED
4560 020564 000414          BR      SPRNTA ;BR TO COMMON SECTION.
4561
4562 020566 010267 160326      SPRNT1: MOV      R2,      $GDADR ;GET THE MEMORY ADDRESS BEING TESTED
4563 020572 005367 160322      DEC      $GDADR ;ADJUST IT FOR PRINTOUT.
4564 020576 000407          BR      SPRNTA ;BR TO COMMON SECTION.
4565
4566 020600 010367 160352      SPRNT3: MOV      R3,      $TMPD ;GET THE DATA IN R3.
4567 020604 010267 160310      SPRNT2: MOV      R2,      $GDADR ;GET THE MEMORY ADDRESS BEING TESTED
4568 020610 162767 000002 160302      SUB      #2,      $GDADR ;ADJUST IT FOR PRINTOUT.
4569 020616 010067 160302      SPRNTA: MOV      R0,      $GDDAT ;GET WHAT THE DATA SHOULD BE
4570 020622 010167 160300      SPRNTB: MOV      R1,      $BDDAT ;GET WHAT THE DATA WAS
4571 020626 000207          RTS      PC              ;RETURN TO ENTER ERROR ROUTINES
4572
4573 ;*****
4574 ;* SUBROUTINE TO TYPE OUT A MAP OF 4K BANK.
4575 ;* R0 POINTS TO THE MAP UPON ENTERING THIS ROUTINE.

```

JOB

```

4576
4577 020630 005710
4578 020632 001007
4579 020634 005760 000002
4580 020640 001034
4581 020642 004567 002372
4582 020646 025645
4583
4584 020650 000475
4585 020652
4586 020652 010146
4587 020654 010246
4588 020656 010346
4589 020660 010446
4590 020662 012701 000001
4591 020666 005002
4592 020670 012703 177777
4593 020674 010304
4594 020676 030110
4595 020700 001014
4596 020702 030260 000002
4597 020706 001011
4598 020710 105703
4599 020712 001042
4600 020714 162703 000001
4601 020720 005604
4602 020722 004567 002312
4603 020726 025241
4604 020730 000410
4605 020732 105703
4606 020734 001431
4607 020736 062703 000001
4608 020742 005504
4609 020744 004567 002270
4610 020750 025231
4611 020752
4612 020752 010346
4613 020754 010446
4614 020756 006303
4615 020760 006104
4616 020762 006003
4617 020764 010446
4618
4619
4620
4621 020766 013746 177776
4622 020772 004767 003410
4623 020776 003
4624 020777 000
4625 021000 010346
4626
4627
4628
4629 021002 013746 177776
4630 021006 004767 003374
4631 021012 005

*****
TYPMAP: TST (R0) ;CHECK IF ANY MEMORY IN MAP...LO 64K.
        BNE 1$ ;BR IF MEMORY IN MAP.
        TST 2(R0) ;...HI 64K.
        BNE 1$ ;BR IF MEMORY IN MAP.
        JSR R5, $SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
        .WORD NOMEM ;ADDRESS OF MESSAGE TO BE TYPED
        ;"NO MEMORY FOUND."
        BR 6$ ;EXIT
1$:
        MOV R1,-(SP) ;:PUSH R1 ON STACK
        MOV R2,-(SP) ;:PUSH R2 ON STACK
        MOV R3,-(SP) ;:PUSH R3 ON STACK
        MOV R4,-(SP) ;:PUSH R4 ON STACK
        MOV #BIT0, R1 ;SET UP BANK POINTER...LO 64K.
        CLR R2 ;HI 64K.
        MOV #-1, R3 ;SET UP ADDRESS POINTER TO -1.
        MOV R3, R4 ;HI BITS OF ADDRESS AS WILL.
2$:
        BIT R1, (R0) ;CHECK THE MAP FOR THIS BANK.
        BNE 3$ ;BR IF THIS BANK PRESENT.
        BIT R2, 2(R0) ;CHECK HI 64K MAP.
        BNE 3$ ;BR IF THIS BANK PRESENT.
        TSTB R3 ;CHECK FOR PREVIOUS PRINTOUT.
        BNE 5$ ;BR IF ALREADY TYPED "TO"
        SUB #1, R3 ;BACK UP TO LAST ADR OF PREVIOUS BANK.
        SBC R4 ;...HI ADDRESS BITS.
        JSR R5, $SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
        .WORD TO ;ADDRESS OF MESSAGE TO BE TYPED
        BR 4$ ;GO TO TYPE THE ADDRESS.
3$:
        TSTB R3 ;CHECK FOR PREVIOUS TYPEOUT.
        BEQ 5$ ;BR IF ALREADY TYPE "FROM".
        ADD #1, R3 ;POINT TO FIRST ADDRESS OF THIS BANK.
        ADC R4 ;...HI BITS OF ADDRESS.
        JSR R5, $SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
        .WORD FROM ;ADDRESS OF MESSAGE TO BE TYPED
4$:
        MOV R3,-(SP) ;:PUSH R3 ON STACK
        MOV R4,-(SP) ;:PUSH R4 ON STACK
        ASL R3 ;BIT 15 INTO C-BIT
        ROL R4 ;BIT 15 INTO R4.
        ROR R3 ;RESTORE BITS 14-0.
        MOV R4,-(SP) ;:SAVE R4 FOR TYPEOUT
        ;:TYPE ADDRESS BITS 21-15
;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOS ROUTINE
;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
        MOV @#PSW, -(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
        JSR PC, $TYPOS ;GO TO THE SUBROUTINE
        .BYTE 3 ;:TYPE 3 DIGIT(S)
        .BYTE 0 ;:SUPPRESS LEADING ZEROS
        MOV R3,-(SP) ;:SAVE R3 FOR TYPEOUT
        ;:TYPE ADDRESS BITS 14-0
;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOS ROUTINE
;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
        MOV @#PSW, -(SP) ;PUT THE PROCESSOR STATJS ON THE STACK
        JSR PC, $TYPOS ;GO TO THE SUBROUTINE
        .BYTE 5 ;:TYPE 5 DIGIT(S)

```

K08

```

4632 021013 001 .BYTE 1 ;:TYPE LEADING ZEROS
4633 021014 012604 MOV (SP)+,R4 ;:POP STACK INTO R4
4634 021016 012603 MOV (SP)+,R3 ;:POP STACK INTO R3
4635 021020 062703 020000 5$: ADD #20000, R3 ;:UPDATE TO NEXT BANK.
4636 021024 005504 ADC R4 ;:HI ADDRESS BITS.
4637 021026 006301 ASL R1 ;:SHIFT POINTER...LO 64K.
4638 021030 006102 ROL R2 ;:HI 64K.
4639 021032 103321 BCC 2$ ;:BR IF MORE BANKS.
4640 021034 012604 MOV (SP)+,R4 ;:POP STACK INTO R4
4641 021036 012603 MOV (SP)+,R3 ;:POP STACK INTO R3
4642 021040 012602 MOV (SP)+,R2 ;:POP STACK INTO R2
4643 021042 012601 MOV (SP)+,R1 ;:POP STACK INTO R1
4644 021044 000207 6$: RTS PC ;:RETURN.
4645
4646 ;*****
4647
4648 .SBTTL SCOPE HANDLER ROUTINE
4649
4650 ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
4651 ;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
4652 ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
4653 ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
4654 ;*SW14=1 LOOP ON TEST
4655 ;*SW11=1 INHIBIT ITERATIONS
4656 ;*SW09=1 LOOP ON ERROR
4657 ;*SW08=1 LOOP ON TEST IN SWR<4:0>
4658 ;*CALL
4659 ;* SCOPE ;:SCOPE=IOT
4660
4661 021046 $$SCOPE:
4662 021046 012504 MOV (R5)+, R4 ;:SAVE MINIMUM BLOCK MASK NEXT TEST.
4663 021050 010516 MOV R5, (SP) ;:PUT RETURN PC ONTO STACK, SIMULATE JSR PC.
4664 021052 032777 040000 160056 1$: BIT #BIT14, $SWR ;:LOOP ON PRESENT TEST?
4665 021060 001117 BNE $OVER ;:YES IF SW14=1
4666 ;*****START OF CODE FOR THE XOR TESTER*****
4667 021062 000416 $XTSTR: BR 6$ ;:IF RUNNING ON THE "XOR" TESTER CHANGE
4668 ;:THIS INSTRUCTION TO A "NOP" (NOP=240)
4669 021064 013746 000004 MOV @#ERRVEC, -(SP) ;:SAVE THE CONTENTS OF THE ERROR VECTOR
4670 021070 012737 021110 000004 MOV #$$, @#ERRVEC ;:SET FOR TIMEOUT
4671 021076 005737 177060 TST @#177060 ;:TIME OUT ON XOR?
4672 021102 012637 000004 MOV (SP)+, @#ERRVEC ;:RESTORE THE ERROR VECTOR
4673 021106 000466 BR $$VLAD ;:GO TO THE NEXT TEST
4674 021110 022626 5$: CMP (SP)+, (SP)+ ;:CLEAR THE STACK AFTER A TIME OUT
4675 021112 012637 000004 MOV (SP)+, @#ERRVEC ;:RESTORE THE ERROR VECTOR
4676 021116 000426 BR 7$ ;:LOOP ON THE PRESENT TEST
4677 021120 6$; *****END OF CODE FOR THE XOR TESTER*****
4678 021120 032777 000400 160010 BIT #BIT09, $SWR ;:LOOP ON SPEC. TEST?
4679 021126 001407 BEQ 2$ ;:BR IF NO
4680 021130 017746 160002 MOV @SWR, -(SP) ;:SET DESIRED TEST NUM. FROM SWR
4681 021134 042716 000340 BIC #$$SWRMK, (SP) ;:STRIP AWAY UNDESIRED BITS
4682 021140 122667 157736 CMPB (SP)+, $TSTNM ;:ON THE RIGHT TEST?
4683 021144 001465 BEQ $OVER ;:BR IF YES
4684 021146 105767 157731 2$: TSTB $ERFLG ;:HAS AN ERROR OCCURRED?
4685 021152 001421 BEQ 3$ ;:BR IF NO
4686 021154 126767 157735 157721 CMPB $ERMAX, $ERFLG ;:MAX. ERRORS FOR THIS TEST OCCURRED?
4687 021162 101015 BHI 3$ ;:BR IF NO
  
```

4688	021164	032777	001000	157744		BIT	#BIT09, QSWR	:: LOOP ON ERROR?
4689	021172	001404				BEQ	4\$:: BR IF NO
4690	021174	016767	157710	157704	7\$:	MOV	\$LPERR, \$LPADR	:: SET LOOP ADDRESS TO LAST SCOPE
4691	021202	000446				BR	\$OVER	
4692	021204	105067	157673		4\$:	CLRB	\$ERFLG	:: ZERO THE ERROR FLAG
4693	021210	005067	157752			CLR	\$TIMES	:: CLEAR THE NUMBER OF ITERATIONS TO MAKE
4694	021214	000415				SR	1\$:: ESCAPE TO THE NEXT TEST
4695	021216	032777	004000	157712	3\$:	BIT	#BIT11, QSWR	:: INHIBIT ITERATIONS?
4696	021224	001011				BNE	1\$:: BR IF YES
4697	021226	005767	157756			TST	\$PASS	:: IF FIRST PASS OF PROGRAM
4698	021232	001406				BEQ	1\$:: INHIBIT ITERATIONS
4699	021234	005267	157644			INC	\$ICNT	:: INCREMENT ITERATION COUNT
4700	021240	026767	157722	157636		CMP	\$TIMES, \$ICNT	:: CHECK THE NUMBER OF ITERATIONS MADE
4701	021246	002024				BGE	\$OVER	:: BR IF MORE ITERATION REQUIRED
4702	021250	012767	000001	157626	1\$:	MOV	#1, \$ICNT	:: REINITIALIZE THE ITERATION COUNTER
4703	021256	016767	000552	157702		MOV	\$MXCNT, \$TIMES	:: SET NUMBER OF ITERATIONS TO DO
4704	021264	105267	157612		\$SVLAD:	INCB	\$TSTNM	:: COUNT TEST NUMBERS
4705	021270	116767	157606	157710		MOVB	\$TSTNM, \$TESTN	:: SET TEST NUMBER IN APT MAILBOX
4706	021276	011667	157604			MOV	(SP), \$LPADR	:: SAVE SCOPE LOOP ADDRESS
4707	021302	011667	157602			MOV	(SP), \$LPERR	:: SAVE ERROR LOOP ADDRESS
4708	021306	005067	157656			CLR	\$ESCAPE	:: CLEAR THE ESCAPE FROM ERROR ADDRESS
4709	021312	112767	000001	157575		MOVB	#1, \$ERMAX	:: ONLY ALLOW ONE(1) ERROR ON NEXT TEST
4710	021320	016777	157556	157612	\$OVER:	MOV	\$TSTNM, QDISPLAY	:: DISPLAY TEST NUMBER
4711	021326	016716	157554			MOV	\$LPADR, (SP)	:: FUDGE RETURN ADDRESS
4712	021332	020516			INSERT:	CMP	R5, (SP)	:: CHECK FOR LOOP ON TEST.
4713	021334	001402				BEQ	1\$:: BR IF START NEXT TEST.
4714	021336	000167	000470			JMP	ENDINS	:: JMP IF LOOP ON LAST TEST.
4715	021342	012767	037777	160234	1\$:	MOV	#37777, BLKMSK	:: SET BK BOUNDRY MASK.
4716	021350	005767	157634			TST	\$PASS	:: CHECK FOR PASS 0.
4717	021354	001404				BEQ	2\$:: BR IF PASS 0
4718	021356	126727	157520	000023		CMPB	\$TSTNM, #23	:: CHECK IF IN SECTION 3.
4719	021364	103002				BHIS	3\$:: BR IF IN SECTION 3.
4720	021366	006267	160212		2\$:	ASR	BLKMSK	:: RESET BOUNDRY TO 4K.
4721	021372	016767	160162	160162	3\$:	MOV	FSTADR, TMPFAD	:: GET FIRST ADDRESS.
4722	021400	005767	157174			TST	RELOC	:: CHECK IF PRG RELOCATED.
4723	021404	001430				BEQ	4\$:: BR IF NOT RELOCATED.
4724	021406	032777	000040	157522		BIT	#SW05, QSWR	:: CHECK IF LOC 0-776 TO BE PROTECTED.
4725	021414	001424				BEQ	4\$:: BR IF SW NOT SET.
4726	021416	026727	160140	001000		CMP	TMPFAD, #1000	:: CHECK IF NOT BEING TESTED.
4727	021424	103020				BHIS	4\$:: BR IF ALREADY PROTECTED.
4728	021426	012767	001000	160126		MOV	#1000, TMPFAD	:: RESET FIRST ADDRESS.
4729	021434	052767	000001	160124		BIS	#BIT0, FADMAP	:: SET FLAG IN FIRST BANK.
4730	021442	026727	160124	001000		CMP	LSTADR, #1000	:: CHECK IF GONE PAST LAST ADR.
4731	021450	101006				BHI	4\$:: BR IF ENOUGH MEMORY.
4732	021452	004567	001562			JSR	R5, \$PRINT	:: GO PRINT OUT THE FOLLOWING MESSAGE.
4733	021456	026316				.WORD	NOBTST	:: ADDRESS OF MESSAGE TO BE TYPED
4734								:: "NO MEMORY TESTED"
4735	021460	016716	160154			MOV	.TST32, (SP)	:: ADJUST RETURN ADR FOR ABORT.
4736	021464	000207				RTS	PC	:: ABORT.
4737	021466	016767	160100	160100	4\$:	MOV	LSTADR, TEMPLAD	:: GET LAST ADDRESS.
4738	021474	016767	160032	160024		MOV	SAVTST, TSTMAP	:: GET TEST MAP, LC 64K.
4739	021502	016767	160026	160020		MOV	SAVTST+2, TSTMAP+2	:: HI 64K.
4740	021510	046767	157066	160010		BIC	PRGMAP, TSTMAP	:: DON'T TEST OVER THE PROGRAM.
4741	021516	046767	157062	160004		BIC	PRGMAP+2, TSTMAP+2	
4742	021524	005767	157460			TST	\$PASS	:: CHECK FOR FIRST PASS
4743	021530	001011				BNE	10\$:: BR IF NOT FIRST PASS.

4744	021532	032767	000003	157766		BIT	#3,	TSTMAP	;CHECK IF FIRST TWO BANKS AVA' GALE.
4745	021540	001405				BEQ	10\$;NOT TESTING FIRST 2 BANKS.
4746	021542	042767	177774	157756		BIC	#177774,	TSTMAP	;CLR ALL BUT FIRST 2 BANKS.
4747	021550	005067	157754			CLR	TSTMAP+2		
4748	021554	005704			10\$:	TST	R4		;CHECK FOR A MINIMUM BLOCK SIZE.
4749	021556	001503				BEQ	20\$;BR IF NO MIN BLOCK SIZE.
4750	021560	030467	157776			BIT	R4,	TMPFAD	;CHECK IF FIRST ADR ON BLOCK BOUNDARY.
4751	021564	001416				BEQ	11\$;BR IF FIRST ADR ON BLOCK BOUNDARY.
4752	021566	050467	157770			BIS	R4,	TMPFAD	;ADJUST FIRST ADR TO END OF BLOCK.
4753	021572	005267	157764			INC	TMPFAD		;FIRST ADR TO FIRST ADR OF NEXT BLOCK.
4754	021576	032767	017777	157756		BIT	#MASK4K,	TMPFAD	;CHECK IF FIRST ADR REACHED 4K BOUNDARY.
4755	021604	001006				BNE	11\$;BR IF NOT ON 4K BOUNDARY.
4756	021606	046767	157754	157712		SIC	FADMAP,	TSTMAP	;DON'T TEST FIRST BANK.
4757	021614	046767	157750	157706		BIC	FADMAP+2,	TSTMAP+2	
4758	021622	030467	157746		11\$:	BIT	R4,	TMPFAD	;CHECK IF LAST ADR ON BLOCK BOUNDARY.
4759	021626	001414				BEQ	12\$;BR IF ON BLOCK BOUNDARY.
4760	021630	040467	157740			BIC	R4,	TMPFAD	;ADJUST LAST ADR DOWN TO NEXT BLOCK BOUNDARY.
4761	021634	032767	017777	157732		BIT	#MASK4K,	TMPFAD	;CHECK IF ADJUSTED TO 4K BOUNDARY.
4762	021642	001006				BNE	12\$;BR IF NOT ON 4K BOUNDARY.
4763	021644	046767	157730	157654		BIC	LADMAP,	TSTMAP	;SKIP TESTING LAST BANK.
4764	021652	046767	157734	157650		BIC	LADMAP+2,	TSTMAP+2	
4765	021660	036767	157702	157712	12\$:	BIT	FADMAP,	LADMAP	;CHECK IF FIRST AND LAST IN SAME BANK.
4766	021666	001004				BNE	13\$;BR IF IN SAME BANK.
4767	021670	036767	157674	157704		BIT	FADMAP+2,	LADMAP+2	;... UPPER 64K.
4768	021676	001404				BEQ	14\$;BR IF FIRST AND LAST NOT SAME BANK.
4769	021700	026767	157670	157654	13\$:	CMP	TMPFAD,	TMPFAD	;CHECK IF ANY MEMORY LEFT.
4770	021706	101406				BLOS	15\$;BR IF NO MEMORY TO TEST.
4771	021710	005767	157612		14\$:	TST	TSTMAP		;CHECK IF ANY BANKS LEFT TO TEST!!
4772	021714	001017				BNE	16\$;BR IF TEST MAP NOT EMPTY.
4773	021716	005767	157606			TST	TSTMAP+2		;CHECK FOR ANY BANKS.
4774	021722	001014				BNE	16\$;BR IF TEST MAP NOT EMPTY.
4775	021724				15\$:				
4776	021724	004567	001310			JSR	R5,	SPRINT	;GO PRINT OUT THE FOLLOWING MESSAGE.
4777	021730	026342				.WORD	SKPMES		;ADDRESS OF MESSAGE TO BE TYPED
4778									; "SKIPPING TEST #"
4779	021732	005046				CLR	-(SP)		;CLEAR THE WORD ON THE STACK.
4780	021734	116716	157142			MOVB	\$TSTNM,	(SP)	;PUT THE DATA ON THE STACK.
4781									; * THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE \$TYPOS ROUTINE
4782									; * WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSTEMAC**.
4783	021740	013746	177776			MOV	\$PSW,	-(SP)	;PUT THE PROCESSOR STATUS ON THE STACK
4784	021744	004767	002436			JSR	PC,	\$TYPOS	;GO TO THE SUBROUTINE
4785	021750	003				.BYTE	3		;TYPE 3 DIGITS.
4786	021751	001				.BYTE	1		;TYPE LEADING ZEROS.
4787	021752	000427				BR	ENDINS		;RETURN TO SKIP TEST.
4788	021754	062716	000004	157120	16\$:	ADD	#4,	(SP)	;SKIP THE SKIP ON RETURN.
4789	021760	062767	000004	157570	20\$:	ADD	#4,	\$LPADR	;ADJUST THE LOOP ADR PAST THE SKIP.
4790	021766	012767	017777			MOV	#MASK4K,	FADMSK	;GET 4K MASK.
4791	021774	016705	157562			MOV	TMPFAD,	R5	;GET FIRST ADR.
4792	022000	040567	157560		21\$:	BIC	R5,	FADMSK	;CLR MASK ABOVE LOWEST BIT OF FIRST ADR.
4793	022004	006305				ASL	R5		;MOVE LOWEST BIT UP ONE.
4794	022006	001374				BNE	21\$;LOOP UNTIL OVERFLOW.
4795	022010	012767	017777	157560		MOV	#MASK4K,	LADMSK	;SET MASK BITS
4796	022016	016705	157552			MOV	TMPFAD,	R5	;GET LAST ADR.
4797	022022	040567	157550		22\$:	BIC	R5,	LADMSK	;CLR ALL MASK BITS ABOVE LOWEST BIT IN LAST ADR.
4798	022026	006305				ASL	R5		;MOVE LOWEST BIT OF LAST ADR UP ONE.
4799	022030	001374				BNE	22\$;LOOP UNTIL OVERFLW.


```

4800 022032 000207      ENDINS: R1S      PC      ;EXIT SCOPE ROUTINE BACK TO TEST.
4801 022034 000004      $MXCNT: 4      ;MAX. NUMBER OF ITERATIONS
4802      ;*****
4803
4804      .SBTTL  ERROR HANDLER ROUTINE
4805
4806      ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
4807      ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
4808      ;*AND GO TO $ERRTYP ON ERROR
4809      ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
4810      ;*SW15=1      HALT ON ERROR
4811      ;*SW13=1      INHIBIT ERROR TYPEOUTS
4812      ;*SW10=1      BELL ON ERROR
4813      ;*SW09=1      LOOP ON ERROR
4814      ;*CALL
4815      ;*      ERROR      N      ;;ERROR=EMT AND N=ERROR ITEM NUMBER
4816
4817      $ERROR:
4818 022036 062716 000002      ADD      #2      (SP)      ;ADJUST POINTER PAST CODE WORD.
4819 022042 105267 157035      7$:      INCB      $ERFLG      ;SET THE ERROR FLAG
4820 022046 001775      BEQ      7$      ;DON'T LET THE FLAG GO TO ZERO
4821 022050 016777 157026 157062      MOV      $TSTNM,$DISPLAY ;DISPLAY TEST NUMBER AND ERROR FLAG
4822 022056 032777 002000 157052      BIT      #BIT10,$SWR      ;BELL ON ERROR?
4823 022064 001403      BEQ      1$      ;NO - SKIP
4824 022066 004567 001146      JSR      R5      $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
4825 022072 001172      .WORD    $BELL      ;ADDRESS OF MESSAGE TO BE TYPED
4826 022074 005267 157012      1$:      INC      $ERTTL      ;COUNT THE NUMBER OF ERRORS
4827 022100 011667 157012      MOV      (SP), $ERRPC ;GET ADDRESS OF ERROR INSTRUCTION
4828 022104 162767 000002 157004      SUB      #2, $ERRPC
4829 022112 117767 157000 156774      MOVB    $ERRPC, $ITEMB ;STRIP AND SAVE THE ERROR ITEM CODE
4830 022120 032777 020000 157010      BIT      #BIT13,$SWR ;SKIP TYPEOUT IF SET
4831 022126 001005      BNE      20$      ;SKIP TYPEOUTS
4832 022130 004767 000074      JSR      PC, $ERRTYP ;GO TO USER ERROR ROUTINE
4833 022134 004567 001100      JSR      R5,      $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
4834 022140 001177      .WORD    $CRLF      ;ADDRESS OF MESSAGE TO BE TYPED
4835 022142
4836 022142 122767 000001 157052      20$:     CMPB    #APTENV, $ENV ;RUNNING IN APT MODE
4837 022150 001007      BNE      2$      ;NO SKIP APT ERROR REPORT
4838 022152 116767 156736 000004      MOVB    $ITEMB, 21$ ;SET ITEM NUMBER AS ERROR NUMBER
4839 022160 004767 001400      JSR      PC, $ATY4 ;REPORT FATAL ERROR TO APT
4840 022164 000      21$:     .BYTE  0
4841 022165 000      .BYTE  0
4842 022166 000777      22$:     BR      22$ ;APT ERROR LOOP
4843 022170 005777 156742      2$:     TST    $SWR ;HALT ON ERROR
4844 022174 100001      BPL      3$      ;SKIP IF CONTINUE
4845 022176 000000      HALT ;HALT ON ERROR!
4846 022200 032777 001000 156730      3$:     BIT    #BIT09,$SWR ;LOOP ON ERROR SWITCH SET?
4847 022206 001402      BEQ      4$      ;BR IF NO
4848 022210 016716 156674      MOV      $LPERR, (SP) ;FUDGE RETURN FOR LOOPING
4849 022214 005767 156750      4$:     TST    $ESCAPE ;CHECK FOR AN ESCAPE ADDRESS
4850 022220 001402      BEQ      5$      ;BR IF NONE
4851 022222 016716 156742      MOV      $ESCAPE, (SP) ;FUDGE RETURN ADDRESS FOR ESCAPE
4852 022226
4853 022226 000207      5$:     RTS      PC
4854      ;*****
4855

```

.SBTTL ERROR MESSAGE TYPEOUT ROUTINE

:*THIS ROUTINE USES THE "ITEM CONTROL BYTE" (\$ITEMB) TO DETERMINE WHICH
:*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" (\$ERRTB),
:*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.

\$ERRTYP:

0022230
0022230 004567 001004
0022230 001177
0022236 010046
0022240 005000
0022242 156700 156646
0022246 001207

0222250 016746 156642

0222254 013746 177776
0222260 004767 002146
0222264 000513
0222266 016767 156624 157216
0222274 166767 156300 157210
0222302 005300
0222304 006300
0222306 006300
0222310 006300
0222312 066700 157316
0222316 012067 000006
0222322 001406
0222324 004567 000710
0222330 000000
0222332 004567 000702
0222336 001177
0222340 012067 000006
0222344 001406
0222346 004567 000666
0222352 000000
0222354 004567 000660
0222360 001177
0222362 010146
0222364 012001
0222366 001451
0222370 066701 156204
0222374 012000
0222376 066700 156176
0222402 105720
0222404 001006
0222406 013146

022410 013746 177776
022414 004767 002012
022420 000426
022422 100426
022424 013146

```
JSR R5, SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
;WORD $R0LFB ;ADDRESS OF MESSAGE TO BE TYPED
MOV RO, -(SP) ;SAVE RO
CLR RO ;PICKUP THE ITEM INDEX
B:59 $ITEMB, RO ;IF ITEM NUMBER IS ZERO, JUST
SNE 1$ ;TYPE THE PC OF THE ERROR
MOV $ERRPC, -(SP) ;;SAVE $ERRPC FOR TYPEOUT
;;ERROR ADDRESS
:* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOC ROUTINE
:* WITHOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
MOV @PSW, -(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
JSR PC, $TYPOC ;GO TO THE SUBROUTINE
BR 10$ ;GET OUT
MOV $ERRPC, $VERPC ;SET UP VIRTUAL PC FOR TYPEOUT.
SUB RELOC, $VERPC ;MAKE VIRTUAL IF NOT ALREADY.
DEC RO ;ADJUST THE INDEX SO THAT IT WILL
ASL RO ; WORK FOR THE ERROR TABLE
ASL RO
ASL RO
ADD $ERRTB, RO ;FORM TABLE POINTER
MOV (RO)+, 2$ ;PICKUP "ERROR MESSAGE" POINTER
BEQ 3$ ;SKIP TYPEOUT IF NO POINTER
JSR R5, SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
;WORD 0 ;"ERROR MESSAGE" POINTER GOES HERE
JSR R5, SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
;WORD $R0LFB ;ADDRESS OF MESSAGE TO BE TYPED
MOV (RO)+, 4$ ;PICKUP "DATA HEADER" POINTER
BEQ 5$ ;SKIP TYPEOUT IF 0
JSR R5, SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
;WORD 0 ;"DATA HEADER" POINTER GOES HERE
JSR R5, SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
;WORD $R0LFB ;ADDRESS OF MESSAGE TO BE TYPED
MOV R1, -(SP) ;SAVE R1
MOV (RO)+, R1 ;PICKUP "DATA TABLE" POINTER
BEQ 9$ ;BR IF NO DATA TO BE TYPED
ADD RELOC, R1 ;ADJUST POINTER
MOV (RO)+, R0 ;PICKUP "DATA FORMAT" POINTER
ADD RELOC, R0 ;ADJUST POINTER.
TSTB (RO)+ ;CHECK THE FORMAT
BNE 7$ ;BR IF NOT 16-BIT OCTAL
MOV @R1, -(SP) ;;SAVE @R1+ FOR TYPEOUT
:* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOC ROUTINE
:* WITHOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
MOV @PSW, -(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
JSR PC, $TYPOC ;GO TO THE SUBROUTINE
BR 8$
BNI 17$ ;BRANCH IF NOT DECIMAL
MOV @R1, -(SP) ;;SAVE @R1+ FOR TYPEOUT
```

```

4912      : * THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPCS ROUTINE
4913      : * WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
4914      MOV      2(PSW), -(SP)      ; PUT THE PROCESSOR STATUS ON THE STACK
4915      JSR      PC, $TYPCS        ; GO TO THE SUBROUTINE
4916      BR      9$                ; SKIP
4917      : *
4918      17$:    CMPB   #-1, -1(R0)   ; CHECK FOR 18-BIT ADDRESS FORMAT.
4919      SNE     18$                ; BR IF NOT 18-BIT ADDRESS FORMAT.
4920      MOV     2(R1)+, -(SP)       ; PUT THE DATA ON THE STACK.
4921      JSR     PC, $TYPAD        ; DETERMINE THE PHYSICAL ADDRESS AND TYPE IT.
4922      BR     8$                ; SKIP
4923      18$:
4924      CLR     -(SP)              ; CLEAR THE WORD ON THE STACK.
4925      MOVB   2(R1)+, (SP)        ; PUT THE DATA ON THE STACK.
4926      : * THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOS ROUTINE
4927      : * WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
4928      MOV     2(PSW), -(SP)      ; PUT THE PROCESSOR STATUS ON THE STACK
4929      JSR     PC, $TYPOS        ; GO TO THE SUBROUTINE
4930      .BYTE  3                    ; TYPE 3 DIGITS.
4931      .BYTE  1                    ; TYPE LEADING ZEROS.
4932      9$:    TST     (R1)          ; IS THERE ANOTHER NUMBER?
4933      BEQ     9$                ; BR IF NO
4934      JSR     R5, $SPRINT        ; GO PRINT OUT THE FOLLOWING MESSAGE.
4935      .WORD  11$                ; ADDRESS OF MESSAGE TO BE TYPED
4936      BR     6$                ; LOOP
4937      9$:    MOV     (SP)+, R1     ; RESTORE R1
4938      10$:   MOV     (SP)+, R0     ; RESTORE R0
4939      JSR     R5, $SPRINT        ; GO PRINT OUT THE FOLLOWING MESSAGE.
4940      .WORD  $CALF              ; ADDRESS OF MESSAGE TO BE TYPED
4941      RTS     PC                ; RETURN
4942      11$:   .ASCIZ  /           ; TAB CHARACTER.
4943      .EVEN
4944      : *****
4945      .SBTTL  TTY INPUT ROUTINE
4946      : * THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
4947      : * CALL:
4948      : *   RDCHR                ; INPUT A SINGLE CHARACTER FROM THE TTY
4949      : *   RETURN HERE          ; CHARACTER IS ON THE STACK
4950      :
4951      $RDCHR: MOV     (SP), -(SP)   ; PUSH DOWN THE PC
4952      MOV     4(SP), 2(SP)         ; SAVE THE PS
4953      1$:    TSTB   2(STK)         ; WAIT FOR
4954      BPL     1$                  ; A CHARACTER
4955      MOVB   2(STKB, 4(SP))        ; READ THE TTY
4956      BIC     8(C<17>), 4(SP)     ; GET RID OF JUNK IF ANY
4957      RTI
4958      : *****
4959      : * THIS ROUTINE WILL INPUT A STRING FROM THE TTY
4960      : * CALL:
4961      : *   RDLIN                ; INPUT A STRING FROM THE TTY
4962      : *   RETURN HERE          ; ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
4963      : *   TERMINATOR WILL BE A BYTE OF ALL 0'S

```

4968	022564	010346			SRDLIN: MOV R3, -(SP) ;:SAVE R3
4969	022566	005046			CLR -(SP) ;:CLEAR THE RUBOUT KEY
4970	022568	012703	023055	1S:	MOV #TTYIN,R3 ;:GET ADDRESS
4971	022574	022703	023065	2S:	CMP #TTYIN+8.,R3 ;:BUFFER FULL?
4972	022600	101467			BLOS 4S ;:BR IF YES
4973					;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE SRDCHR ROUTINE
4974					;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
4975	022602	013746	177776		MOV #PSW, -(SP) ;:PUT THE PROCESSOR STATUS ON THE STACK
4976	022606	004767	177716		JSR PC, \$R0CHR ;:GO TO THE SUBROUTINE
4977	022612	112613			MOVB (SP)+,(R3) ;:GET CHARACTER
4978	022614	122713	000177		CMPB #177,(R3) ;:IS IT A RUBOUT
4979	022620	001024			BNE 5S ;:BR IF NO
4980	022622	005716			TST (SP) ;:IS THIS THE FIRST RUBOUT?
4981	022624	001010			BNE 6S ;:BR IF NO
4982	022626	112767	000134	000212	MOVB #' \,9S ;:TYPE A BACK SLASH
4983	022634	004567	000400		JSR R5, \$PRINT ;:GO PRINT OUT THE FOLLOWING MESSAGE.
4984	022640	023046			.WORD 9S ;:ADDRESS OF MESSAGE TO BE TYPED
4985	022642	012716	177777		MOV #-1,(SP) ;:SET THE RUBOUT KEY
4986	022646	005303		6S:	DEC R3 ;:BACKUP BY ONE
4987	022650	020327	023055		CMP R3,#TTYIN ;:STACK EMPTY?
4988	022654	103441			BLO 4S ;:BR IF YES
4989	022656	111367	000164		MOVB (R3),9S ;:SETUP TO TYPEOUT THE DELETED CHAR.
4990	022662	004567	000352		JSR R5, \$PRINT ;:GO PRINT OUT THE FOLLOWING MESSAGE.
4991	022666	023046			.WORD 9S ;:ADDRESS OF MESSAGE TO BE TYPED
4992	022670	000741			BR 2S ;:GO READ ANOTHER CHAR.
4993	022672	005716		5S:	TST (SP) ;:RUBOUT KEY SET?
4994	022674	001407			BEQ 7S ;:BR IF NO
4995	022676	112767	000134	000142	MOVB #' \,9S ;:TYPE A BACK SLASH
4996	022704	004567	000330		JSR R5, \$PRINT ;:GO PRINT OUT THE FOLLOWING MESSAGE.
4997	022710	023046			.WORD 9S ;:ADDRESS OF MESSAGE TO BE TYPED
4998	022712	005016			CLR (SP) ;:CLEAR THE RUBOUT KEY
4999	022714	122713	000025	7S:	CMPB #25,(R3) ;:IS CHARACTER A CTRL U?
5000	022720	001004			BNE 8S ;:BR IF NO
5001	022722	004567	000312		JSR R5, \$PRINT ;:GO PRINT OUT THE FOLLOWING MESSAGE.
5002	022726	023050			.WORD \$CNTLJ ;:ADDRESS OF MESSAGE TO BE TYPED
5003	022730	000717			BR 1S ;:GO START OVER
5004	022732	122713	000012	9S:	CMPB #12,(R3) ;:IS CHARACTER A "LF"?
5005	022736	001014			BNE 3S ;:BRANCH IF NO
5006	022740	105013			CLAB (R3) ;:CLEAR THE CHARACTER
5007	022742	004567	000272		JSR R5, \$PRINT ;:GO PRINT OUT THE FOLLOWING MESSAGE.
5008	022746	001177			.WORD \$CRLF ;:ADDRESS OF MESSAGE TO BE TYPED
5009	022750	004567	000264		JSR R5, \$PRINT ;:GO PRINT OUT THE FOLLOWING MESSAGE.
5010	022754	023055			.WORD \$TTYIN ;:ADDRESS OF MESSAGE TO BE TYPED
5011	022756	000706			BR 2S ;:GO PICKUP ANOTHER CHACTER
5012	022760			4S:	
5013	022760	004567	000254		JSR R5, \$PRINT ;:GO PRINT OUT THE FOLLOWING MESSAGE.
5014	022764	001176			.WORD \$QUES ;:ADDRESS OF MESSAGE TO BE TYPED
5015	022766	000700			BR 1S ;:CLEAR THE BUFFER AND LOOP
5016	022770	111367	000052	3S:	MOVB (R3),9S ;:ECHO THE CHARACTER
5017	022774	004567	000240		JSR R5, \$PRINT ;:GO PRINT OUT THE FOLLOWING MESSAGE.
5018	023000	023046			.WORD 9S ;:ADDRESS OF MESSAGE TO BE TYPED
5019	023002	122723	000015		CMPB #15,(R3)+ ;:CHECK FOR RETURN
5020	023006	001272			BNE 2S ;:LOOP IF NOT RETURN
5021	023010	105063	177777		CLAB -(R3) ;:CLEAR RETURN (THE 15)
5022	023014	004567	000220		JSR R5, \$PRINT ;:GO PRINT OUT THE FOLLOWING MESSAGE.
5023	023020	001200			.WORD \$LF ;:ADDRESS OF MESSAGE TO BE TYPED

```

5024 023022 005726          TST      (SP)+          ;; CLEAN RUBOUT KEY FROM THE STACK
5025 023024 012603          MOV      (SP)+,R3      ;; RESTORE R3
5026 023026 011646          MOV      (SP),-(SP)    ;; ADJUST THE STACK AND PUT ADDRESS OF THE
5027 023030 016666 000004 000002  MOV      4(SP),2(SP)    ;; FIRST ASCII CHARACTER ON IT
5028 023036 012756 023055 000004  MOV      *$TTYIN,4(SP)
5029 023044 000002          RTI                    ;; RETURN
5030 023046          000          9$: .BYTE 0          ;; STORAGE FOR ASCII CHAR. TO TYPE
5031 023047          000          .BYTE 0          ;; TERMINATOR
5032 023050 052536 005015 000  $CNTLU: .ASCIZ /?U<<15><<12> ;; CONTROL "U"
5033 023055 000010  $TTYIN: .BLKB 8.          ;; RESERVE 8 BYTES FOR TTY INPUT
5034          023066          .EVEN
5035          ;*****
5036          .SBTTL READ AN OCTAL NUMBER FROM THE TTY
5037
5038          ;*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
5039          ;*CHANGE IT TO BINARY.
5040          ;*THE INPUT CHARACTERS WILL BE CHECKED TO INSURED THEY ARE LEGAL
5041          ;*OCTAL DIGITS. IF AN ILLEGAL CHARACTER IS READ A "?" WILL BE TYPED
5042          ;*FOLLOWED BY A CARRIAGE RETURN-LINE FEED. THE COMPLETE NUMBER MUST
5043          ;*THEN BE RETYPED. THE INPUT IS TERMINATED BY TYPING A CARRIAGE RETURN.
5044          ;*CALL:
5045          ;*
5046          ;*      RDOCT          ;; READ AN OCTAL NUMBER
5047          ;*      RETURN HERE      ;; LOW ORDER BITS ARE ON TOP OF THE STACK
5048          ;*                      ;; HIGH ORDER BITS ARE IN $HI OCT
5049
5050          $RDOCT: MOV      (SP),-(SP)          ;; PROVIDE SPACE FOR THE
5051 023066 011646 000004 000002  MOV      4(SP),2(SP)    ;; INPUT NUMBER
5052 023070 016666          MOV      R0,-(SP)      ;; PUSH R0 ON STACK
5053 023076 010046          MOV      R1,-(SP)      ;; PUSH R1 ON STACK
5054 023100 010146          MOV      R2,-(SP)      ;; PUSH R2 ON STACK
5055 023102 010246
5056          1$:
5057          ;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $RDLIN ROUTINE
5058          ;* WITHOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
5059 023104 013746 177776          MOV      2*PSW, -(SP)  ;; PUT THE PROCESSOR STATUS ON THE STACK
5060 023110 004767 177450          JSR      PC, $RDLIN   ;; GO TO THE SUBROUTINE
5061 023114 012600          MOV      (SP)+,R0     ;; GET ADDRESS OF 1ST CHARACTER
5062 023116 010067 000102  MOV      R0,$$        ;; AND SAVE IT
5063 023122 005001          CLR      R1          ;; CLEAR DATA WORD
5064 023124 005002          CLR      R2
5065 023126 112046          2$: MOV      (R0)+,-(SP)  ;; PICKUP THIS CHARACTER
5066 023130 001420          BEQ      3$          ;; IF ZERO GET OUT
5067 023132 122716 000060  CMP      #'0,(SP)     ;; MAKE SURE THIS CHARACTER
5068 023136 003026          BGT      4$          ;; IS AN OCTAL DIGIT
5069 023140 122716 000067  CMP      #'7,(SP)
5070 023144 002423          BLT      4$
5071 023146 006301          ASL      R1          ;; *2
5072 023150 006102          ROL      R2
5073 023152 006301          ASL      R1          ;; *4
5074 023154 006102          ROL      R2
5075 023156 006301          ASL      R1          ;; *8
5076 023160 006102          ROL      R2
5077 023162 042716 177770  BIC      #'C7,(SP)    ;; STRIP THE ASCII JUNK
5078 023166 062601          ADD      (SP)+,R1     ;; ADD IN THIS DIGIT
5079 023170 000756          BR      2$          ;; LOOP
5080 023172 005726          3$: TST      (SP)+          ;; CLEAN TERMINATOR FROM STACK

```

```

5080 023174 010166 000012
5081 023200 010267 000032
5082 023204 012602
5083 023206 012601
5084 023210 012600
5085 023212 000002
5086 023214 005726
5087 023216 105010
5088 023220 004567 000014
5089 023224 000000
5090 023226 004567 000006
5091 023232 001176
5092 023234 000723
5093 023236 000000
5094
5095
5096
5097
5098
5099
5100 023240 012567 000016
5101 023244 056767 155330 000010
5102
5103
5104 023252 013746 177776
5105 023256 004767 000004
5106 023262 000000
5107 023264 000205
5108
5109
5110
5111
5112
5113
5114
5115
5116
5117
5118
5119
5120
5121
5122
5123
5124
5125
5126
5127 023266 105767 155663
5128 023272 100002
5129 023274 000000
5130 023276 000430
5131 023300 010046
5132 023302 017600 000002
5133 023306 122767 000001 155706
5134 023314 001011
5135 023316 132767 000100 155677

```

```

MOV R1,12(SP) ;;SAVE THE RESULT
MOV R2,$HIOCT
MOV (SP)+,R2 ;;POP STACK INTO R2
MOV (SP)+,R1 ;;POP STACK INTO R1
MOV (SP)+,R0 ;;POP STACK INTO R0
RTI ;;RETURN
4$: TST (SP)+ ;;CLEAN PARTIAL FROM STACK
CLR8 (R0) ;;SET A TERMINATOR
JSR R5, $SPRINT ;;GO PRINT OUT THE FOLLOWING MESSAGE.
5$: .WORD 0
JSR R5, $SPRINT ;;GO PRINT OUT THE FOLLOWING MESSAGE.
.WORD $QUES ;;ADDRESS OF MESSAGE TO BE TYPED
SR 1$ ;;TRY AGAIN
$HIOCT: .WORD 0 ;;HIGH ORDER BITS GO HERE

*****
* SUBROUTINE TO PASS RELOCATED MESSAGE ADDRESSES TO THE $TYPE ROUTINE.
* CALL: JSR R5, $SPRINT
* <MESSAGE VIRTUAL ADDRESS>
*****
$SPRINT: MOV (R5)+, 1$ ;;GET THE MESSAGE VIRTUAL ADDRESS.
ADD RELOC, 1$ ;;MAKE IT PHYSICAL.
* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPE ROUTINE
* WITHOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
MOV 2*PSW, -(SP) ;;PUT THE PROCESSOR STATUS ON THE STACK
JSR PC, $TYPE ;;GO TO THE SUBROUTINE
1$: .WORD 0 ;;CONTAINS THE PHYSICAL MESSAGE ADDRESS.
RTS R5 ;;RETURN.

*****
.SBTTL TYPE ROUTINE

*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
*NOTE1: $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
*NOTE2: $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
*NOTE3: $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
*
*CALL:
*1) USING A TRAP INSTRUCTION
* TYPE ,MESADR ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
*OR
* TYPE
* MESADR
*

$TYPE: TSTB $TPFLG ;;IS THERE A TERMINAL?
BPL 1$ ;;BR IF YES
HALT ;;HALT HERE IF NO TERMINAL
BR 3$ ;;LEAVE
1$: MOV R0, -(SP) ;;SAVE R0
MOV 2(SP), R0 ;;GET ADDRESS OF ASCIZ STRING
CMPB 2APTENV, $ENV ;;RUNNING IN APT MODE
BNE 62$ ;;NO GO CHECK FOR APT CONSOLE
BITB 2APTSPool, $ENVM ;;SPool MESSAGE TO APT

```



```

5192
5193 .SBTTL APT COMMUNICATIONS ROUTINE
5194 023546 112767 000001 000376 $ATY1: MOVB #1,$FFLG ;TO REPORT FATAL ERROR
5195 023554 112767 000001 000365 $ATY3: MOVB #1,$MFLG ;TO TYPE A MESSAGE
5196 023562 000403 BR $ATYC
5197 023564 112767 000001 000360 $ATY4: MOVB #1,$FFLG ;TO ONLY REPORT FATAL ERROR
5198 023572 $ATYC:
5199 023572 010046 MOV RO,-(SP) ;;PUSH RO ON STACK
5200 023574 010146 MOV RI,-(SP) ;;PUSH RI ON STACK
5201 023576 105767 000346 TSTB $MFLG ;SHOULD TYPE A MESSAGE?
5202 023602 001450 BEQ 5$ ;IF NOT: BR
5203 023604 122767 000001 155410 CMPB #APTENV,$ENV ;OPERATING UNDER APT?
5204 023612 001031 BNE 3$ ;IF NOT: BR
5205 023614 132767 000100 155401 BITB #APTSPOOL,$ENVM ;SHOULD SPOOL MESSAGES?
5206 023622 001425 BEQ 3$ ;IF NOT: BR
5207 023624 017600 000004 MOV #4(SP),RO ;GET MESSAGE ADDR.
5208 023630 062766 000002 000004 ADD #2,4(SP) ;BUMP RETURN ADDR.
5209 023636 005767 155340 1$: TST $MSGTYPE ;SEE IF DONE W/ LAST XMISSION?
5210 023642 001375 BNE 1$ ;IF NOT: WAIT
5211 023644 010067 155346 MOV RO,$MSGAD ;PUT ADDR IN MAILBOX
5212 023650 105720 2$: TSTB (RO)+ ;FIND END OF MESSAGE
5213 023652 001376 BNE 2$
5214 023654 166700 155336 SUB $MSGAD,RO ;SUB START OF MESSAGE
5215 023660 006200 ASR RO ;GET MESSAGE LNTH IN WORDS
5216 023662 010067 155332 MOV RO,$MSGLGT ;PUT LENGTH IN MAILBOX
5217 023666 012767 000004 155306 MOV #4,$MSGTYPE ;TELL APT TO TAKE MSG.
5218 023674 000413 BR 5$
5219 023676 017667 000004 000016 3$: MOV #4(SP),4$ ;PUT MSG ADDR IN JSR LINKAGE
5220 023704 062766 000002 000004 ADD #2,4(SP) ;BUMP RETURN ADDRESS
5221 023712 016746 154060 MOV 177776,-(SP) ;;PUSH 177776 ON STACK
5222 023716 004767 177344 JSR PC,$TYPE ;CALL TYPE MACRO
5223 023722 000000 4$: .WORD
5224 023724 5$:
5225 023724 105767 000221 TSTB $LFLG ;SHOULD LOG AN ERROR?
5226 023730 001422 BEQ 10$ ;IF NOT: BR
5227 023732 017600 000004 MOV #4(SP),RO ;GET ERROR #
5228 023736 062766 000002 000004 ADD #2,4(SP) ;BUMP RETURN ADDR.
5229 023744 012701 001342 MOV #ASTAT,RI ;POINT TO TABLE START
5230 023750 005711 6$: TST (RI) ;END OF TABLE?
5231 023752 100404 BMI 8$ ;IF SO: BR
5232 023754 020021 CMP RO,(RI)+ ;PROPER ENTRY?
5233 023756 001406 BEQ 9$ ;IF SO: BR
5234 023760 005721 TST (RI)+ ;MOVE PAST COUNTER WORD
5235 023762 000772 BR 6$ ;KEEP LOOKING
5236 023764 026701 155520 8$: CMP $APTR,RI ;TABLE FULL?
5237 023770 001402 BEQ 10$ ;IF SO: BR -- NO MORE ROOM
5238 023772 010021 MOV RO,(RI)+ ;SET UP NEW ENTRY
5239 023774 005211 9$: INC (RI) ;BUMP ERROR COUNT
5240 023776 105767 000150 10$: TSTB $FFLG ;SHOULD REPORT FATAL ERROR?
5241 024002 001416 BEQ 12$ ;IF NOT: BR
5242 024004 005767 155212 TST $ENV ;RUNNING UNDER APT?
5243 024010 001413 BEQ 12$ ;IF NOT: BR
5244 024012 005767 155164 11$: TST $MSGTYPE ;FINISHED LAST MESSAGE?
5245 024016 001375 BNE 11$ ;IF NOT: WAIT
5246 024020 017667 000004 155156 MOV #4(SP),$FATAL ;GET ERROR #
5247 024026 062766 000002 000004 ADD #2,4(SP) ;BUMP RETURN ADDR.

```



```

5248 024034 005267 155142          INC      $MSGTYPE      ; TELL APT TO TAKE ERROR
5249 024040 105067 000106      12$: CLR      $FFLG      ; CLEAR FATAL FLAG
5250 024044 105067 000101      CLR      $LFLG      ; CLEAR LOG FLAG
5251 024050 105067 000074      CLR      $MFLG      ; CLEAR MESSAGE FLAG
5252 024054 012601          MOV      (SP)+,R1     ; POP STACK INTO R1
5253 024056 012600          MOV      (SP)+,R0     ; POP STACK INTO R0
5254 024060 000207          RTS      PC           ; RETURN
5255
5256 024062          $ATY6:
5257 024062 010046          MOV      R0, -(SP)    ; PUSH R0 ON STACK
5258 024064 016700 155420          MOV      $APTR, R0
5259 024070 162700 001342          SUB      # $ASTAT, R0 ; GET SIZE OF STAT TABLE
5260 024074 005767 155102      1$: TST      $MSGTY    ; SEE IF DONE LAST COMMUNICATION
5261 024100 001375          BNE     IS           ; IF NOT: WAIT
5262 024102 010067 155112          MOV      R0, $MSGLG   ; SET MESSAGE LENGTH
5263 024106 012767 001342 155102      MOV      # $ASTAT, $MSGAD ; SET MESSAGE ADDR.
5264 024114 012767 000002 155060      MOV      #2, $MSGTY   ; TELL APT TO TAKE STATS.
5265 024122 012600          MOV      (SP)+, R0    ; POP STACK INTO R0
5266 024124 000207          RTS      PC           ; RETURN
5267
5268 024126          $ATY7:
5269 024130 012701 001342          MOV      R0, -(SP)    ; PUSH R0 ON STACK
5270 024134 005721          MOV      # $ASTAT, R1 ; GET START OF TABLE
5271 024136 100402      1$: TST      (R1)+     ; END OF TABLE?
5272 024140 005021          BMI     2$           ; IF SO: BR
5273 024142 000774          CLR      (R1)+       ; CLEAR ERROR COUNT
5274 024144          BR      IS           ; KEEP CLEARING
5275
5276 024144 012600      2$: MOV      (SP)+, R0    ; POP STACK INTO R0
5277 024146 000207          RTS      PC           ; RETURN
5278 024150          $MFLG: .BYTE 0      ; MESSG. FLAG
5279 024151          $LFLG: .BYTE 0      ; LOG FLAG
5280 024152          $FFLG: .BYTE 0      ; FATAL FLAG
5281          .EVEN
5282
5283 APTSIZE=200
5284 APTENV=001
5285 APTSPOOL=100
5286 APTCSUP=040
5287 ;*****
5288 .SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
5289
5290 ; *THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
5291 ; *SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
5292 ; *NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
5293 ; *BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
5294 ; *REPLACED WITH SPACES.
5295 ; *CALL:
5296 ; *      MOV      NUM, -(SP)      ; PUT THE BINARY NUMBER ON THE STACK
5297 ; *      TYPDS          ; GO TO THE ROUTINE
5298
5299 $TYPDS:
5300 MOV      R0, -(SP)      ; PUSH R0 ON STACK
5301 MOV      R1, -(SP)      ; PUSH R1 ON STACK
5302 MOV      R2, -(SP)      ; PUSH R2 ON STACK
5303 MOV      R3, -(SP)      ; PUSH R3 ON STACK
5304 MOV      R5, -(SP)      ; PUSH R5 ON STACK

```

5304	024166	012746	020200		MOV	#20200,-(SP)	::SET BLANK SWITCH AND SIGN
5305	024172	016605	000020		MOV	20(SP),R5	::GET THE INPUT NUMBER
5306	024176	100004			BPL	1\$::BR IF INPUT IS POS.
5307	024200	005405			NEG	R5	::MAKE THE BINARY NUMBER POS.
5308	024202	112766	000055	000001	MOV	#'-1(SP)	::MAKE THE ASCII NUMBER NEG.
5309	024210	016700	154364	1\$:	MOV	RELOC, R0	::GET RELOCATION FACTOR.
5310	024214	012703	024376		MOV	#\$DBLK,R3	::SETUP THE OUTPUT POINTER
5311	024220	060003			ADD	R0, R3	::ADD IN RELOCATION FACTOR.
5312	024222	112723	000040		MOV	#',(R3)+	::SET THE FIRST CHARACTER TO A BLANK
5313	024226	005002		2\$:	CLR	R2	::CLEAR THE BCD NUMBER
5314	024230	016001	024366		MOV	\$DTBL(R0),R1	::GET THE CONSTANT
5315	024234	160105		3\$:	SUB	R1,R5	::FORM THIS BCD DIGIT
5316	024236	002402			SLT	4\$::BR IF DONE
5317	024240	005202			INC	R2	::INCREASE THE BCD DIGIT BY 1
5318	024242	000774			BR	3\$	
5319	024244	060105		4\$:	ADD	R1,R5	::ADD BACK THE CONSTANT
5320	024246	005702			TST	R2	::CHECK IF BCD DIGIT=0
5321	024250	001002			BNE	5\$::FALL THROUGH IF 0
5322	024252	105716			TSTB	(SP)	::STILL DOING LEADING 0'S?
5323	024254	100407			BMI	7\$::BR IF YES
5324	024256	106316		5\$:	ASLB	(SP)	::MSD?
5325	024260	103003			BCC	6\$::BR IF NO
5326	024262	116663	000001	.177777	MOV	1(SP),-1(R3)	::YES--SET THE SIGN
5327	024270	052702	000060	6\$:	BIS	#'0,R2	::MAKE THE BCD DIGIT ASCII
5328	024274	052702	000040	7\$:	BIS	#',R2	::MAKE IT A SPACE IF NOT ALREADY A DIGIT
5329	024300	110223			MOV	R2,(R3)+	::PUT THIS CHARACTER IN THE OUTPUT BUFFER
5330	024302	005720			TST	(R0)+	::JUST INCREMENTING
5331	024304	020067	155326		CMP	R0, .EIGHT	::CHECK THE TABLE INDEX
5332	024310	103746			BLO	2\$::GO DO THE NEXT DIGIT
5333	024312	101002			BHI	8\$::GO TO EXIT
5334	024314	010502			MOV	R5,R2	::GET THE LSD
5335	024316	000764			BR	6\$::GO CHANGE TO ASCII
5336	024320	105726		8\$:	TSTB	(SP)+	::WAS THE LSD THE FIRST NON-ZERO?
5337	024322	100003			BPL	9\$::BR IF NO
5338	024324	116663	177777	177776	MOV	-1(SP),-2(R3)	::YES--SET THE SIGN FOR TYPING
5339	024332	105013		9\$:	CLRB	(R3)	::SET THE TERMINATOR
5340	024334	012605			MOV	(SP)+,R5	::POP STACK INTO R5
5341	024336	012603			MOV	(SP)+,R3	::POP STACK INTO R3
5342	024340	012602			MOV	(SP)+,R2	::POP STACK INTO R2
5343	024342	012601			MOV	(SP)+,R1	::POP STACK INTO R1
5344	024344	012600			MOV	(SP)+,R0	::POP STACK INTO R0
5345	024346	004567	176666		JSR	R5, \$PRINT	::GO PRINT OUT THE FOLLOWING MESSAGE.
5346	024352	024376			.WORD	\$DBLK	::ADDRESS OF MESSAGE TO BE TYPED
5347	024354	016666	000002	000004	MOV	2(SP),4(SP)	::ADJUST THE STACK
5348	024362	012616			MOV	(SP)+,(SP)	
5349	024364	000002			RTI		::RETURN TO USER
5350	024366	023420		\$DTBL:	10000.		
5351	024370	001750			1000.		
5352	024372	000144			100.		
5353	024374	000012			10.		
5354	024376	000004		\$DBLK:	.BLKW 4		

.SBTTL BINARY TO OCTAL (ASCII) AND TYPE

;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT

```

5360      ;*OCTAL (ASCII) NUMBER AND TYPE IT.
5361      ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
5362      ;*CALL:
5363      ;*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
5364      ;*      TYPOS      ;;CALL FOR TYPEOUT
5365      ;*      .BYTE    N      ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
5366      ;*      .BYTE    M      ;;M=1 OR 0
5367      ;*      ;;1=TYPE LEADING ZEROS
5368      ;*      ;;0=SUPPRESS LEADING ZEROS
5369      ;*
5370      ;*$TYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
5371      ;*$TYPOS OR $TYPOC
5372      ;*CALL:
5373      ;*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
5374      ;*      TYPON      ;;CALL FOR TYPEOUT
5375      ;*
5376      ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
5377      ;*CALL:
5378      ;*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
5379      ;*      TYPOC      ;;CALL FOR TYPEOUT
5380
5381      024406 017646 000000      $TYPOS: MOV      2(SP),-(SP)      ;;PICKUP THE MODE
5382      024412 116667 000001 000213      MOV      1(SP),$OFILL      ;;LOAD ZERO FILL SWITCH
5383      024420 112667 000211      MOV      (SP)+,$SOMODE+1      ;;NUMBER OF DIGITS TO TYPE
5384      024424 062716 000002      ADD      #2,(SP)      ;;ADJUST RETURN ADDRESS
5385      024430 000406      BR      $TYPON
5386      024432 112767 000001 000173      $TYPOC: MOV      #1,$OFILL      ;;SET THE ZERO FILL SWITCH
5387      024440 112767 000006 000167      MOV      #6,$SOMODE+1      ;;SET FOR SIX(6) DIGITS
5388      024446 112767 000005 000156      $TYPON: MOV      #5,$OCNT      ;;SET THE ITERATION COUNT
5389      024454 010346      MOV      R3,-(SP)      ;;SAVE R3
5390      024456 010446      MOV      R4,-(SP)      ;;SAVE R4
5391      024460 010546      MOV      R5,-(SP)      ;;SAVE R5
5392      024462 116704 000147      MOV      $SOMODE+1,R4      ;;GET THE NUMBER OF DIGITS TO TYPE
5393      024466 005404      NEG      R4
5394      024470 062704 000006      ADD      #6,R4      ;;SUBTRACT IT FOR MAX. ALLOWED
5395      024474 110467 000134      MOV      R4,$SOMODE      ;;SAVE IT FOR USE
5396      024500 116704 000127      MOV      $OFILL,R4      ;;GET THE ZERO FILL SWITCH
5397      024504 016605 000012      MOV      12(SP),R5      ;;PICKUP THE INPUT NUMBER
5398      024510 005003      CLR      R3      ;;CLEAR THE OUTPUT WORD
5399      024512 006105      1$: ROL      R5      ;;ROTATE MSB INTO "C"
5400      024514 000404      BR      3$      ;;GO DO MSB
5401      024516 006105      2$: ROL      R5      ;;FORM THIS DIGIT
5402      024520 006105      ROL      R5
5403      024522 006105      ROL      R5
5404      024524 010503      MOV      R5,R3
5405      024526 006103      3$: ROL      R3      ;;GET LSB OF THIS DIGIT
5406      024530 105367 000100      DECB      $SOMODE      ;;TYPE THIS DIGIT?
5407      024534 100017      BPL      7$      ;;BR IF NO
5408      024536 042703 177770      BIC      #177770,R3      ;;GET RID OF JUNK
5409      024542 001002      BNE      4$      ;;TEST FOR 0
5410      024544 005704      TST      R4      ;;SUPPRESS THIS 0?
5411      024546 001403      BEQ      5$      ;;BR IF YES
5412      024550 005204      4$: INC      R4      ;;DON'T SUPPRESS ANYMORE 0'S
5413      024552 052703 000060      BIS      #'0,R3      ;;MAKE THIS DIGIT ASCII
5414      024556 052703 000040      5$: BIS      #' ,R3      ;;MAKE ASCII IF NOT ALREADY
5415      024562 110367 000042      MOV      R3,8$      ;;SAVE FOR TYPING

```

```

5416 024566 004567 176446      JSR    R5,    $SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
5417 024572 024630          .WORD    9$          ;ADDRESS OF MESSAGE TO BE TYPED
5418 024574 105367 000032      7$:    DEC8    $OCNT    ;COUNT BY 1
5419 024600 003346          BGT     2$          ;BR IF MORE TO DO
5420 024602 002402          BLT     6$          ;BR IF DONE
5421 024604 005204          INC     R4          ;INSURE LAST DIGIT ISN'T A BLANK
5422 024606 000743          BR      2$          ;GO DO THE LAST DIGIT
5423 024610 012605          6$:    MOV     (SP)+,R5 ;RESTORE R5
5424 024612 012604          MOV     (SP)+,R4 ;RESTORE R4
5425 024614 012603          MOV     (SP)+,R3 ;RESTORE R3
5426 024616 016666 000002 000004  MOV     2(SP),4(SP) ;SET THE STACK FOR RETURNING
5427 024624 012616          MOV     (SP)+,(SP)
5428 024626 000002          RTI                    ;;RETURN
5429 024630 000          8$:    .BYTE    0          ;;STORAGE FOR ASCII DIGIT
5430 024631 000          .BYTE    0          ;;TERMINATOR FOR TYPE ROUTINE
5431 024632 000          $OCNT: .BYTE    0          ;;OCTAL DIGIT COUNTER
5432 024633 000          $OFILL: .BYTE    0          ;;ZERO FILL SWITCH
5433 024634 000000          $OMODE: .WORD    0          ;;NUMBER OF DIGITS TO TYPE
5434          .ERROR TRAP SERVICE ROUTINE
5435 024636 005727          ERRTRP: TST    (PC)+ ;CHECK IF PREV TRAP TO 4 REPORTED
5436 024640 000000          1$:    .WORD    0          ;CONTAINS ERROR REPORTED FLAG
5437 024642 001010          BNE     2$          ;BRANCH IF NOT REPORTED
5438 024644 005267 177770          INC     1$          ;SET DOUBLE TRAP FLAG.
5439 024650 011667 154310          MOV     (SP),    $TMP3 ;SAVE THE BAD PC FOR TYP OUT.
5440 024654 004767 175156          JSR    PC,    $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
5441 024660 000031          .WORD    31          ;ERROR TYPE CODE.
5442 024662 000401          BR      3$          ;SKIP HALT
5443 024664 00000C          2$:    HALT          ;ERROR! SECOND TRAP TO 4 OCCURRED
5444          ;BEFORE FIRST WAS PRINTED
5445 024666 005067 177746          3$:    CLR     1$          ;RETURN TO PROGRAM AND TRY TO RECOVER
5446 024672 000002          RTI
5447
5448          .SBTTL PHYSICAL ADDRESS TYPE ROUTINE
5449          ;* ROUTINE TO TYPE A PHYSICAL ADDRESS (18 BITS).
5450          $TYPAD:
5451 024674 010046          MOV     R0,-(SP) ;PUSH R0 ON STACK
5452 024676 010146          MOV     R1,-(SP) ;PUSH R1 ON STACK
5453 024700 010246          MOV     R2,-(SP) ;PUSH R2 ON STACK
5454 024702 010346          MOV     R3,-(SP) ;PUSH R3 ON STACK
5455 024704 016602 000012          MOV     12(SP), R2 ;GET BASE ADDRESS
5456 024710 005003          CLR     R3          ;WORKING & INDEX REGISTER
5457 024712 005767 153670          TST    MMAVA ;CHECK FOR MEM MGMT AVAILABLE
5458 024716 001430          BEQ     1$          ;BRANCH IF NO MEM MGMT
5459 024720 032737 000001 177572          BIT     #1,    @#SRO ;CHECK IF MEM MGMT ENABLED
5460 024726 001424          BEQ     1$          ;BRANCH IF MEM MGMT NOT ENABLED
5461 024730 010201          MOV     R2,    R1 ;COPY VIRTUAL ADR
5462 024732 006101          ROL     R1          ;SHUFFLE BITS 13,14,15 INTO 1,2,3
5463 024734 006101          ROL     R1
5464 024736 006101          ROL     R1
5465 024740 006101          ROL     R1
5466 024742 006101          ROL     R1
5467 024744 042701 177761          BIC     #177761, R1 ;CLR ALL EXCEPT BITS 1,2,3
5468 024750 062701 172340          ADD     #KIPARO, R1 ;SET TO APPROPRIATE PAR
5469 024754 011101          MOV     (R1),    R1 ;GET CONTENTS OF PAR
5470 024756 012700 000006          MOV     #6,    R0 ;SET UP COUNTER
5471 024762 006301          4$:    ASL     R1 ;SHIFT PAR
    
```

```

5472 024764 006103      ROL      R3      ;SAVE OVERFLOW BITS
5473 024766 077003      SOB      R0      ;COUNT SIX SHIFTS
5474 024770 042702 160000  BIC      #150000, R2 ;SAVE BANK BITS
5475 024774 060102      ADD      R1      ;COMPUTE PHYSICAL ADDRESS
5476 024776 005503      ADC      R3      ;MAKE SURE CARRY ISN'T LOST!
5477 025000 006302      1$:     ASL      R2      ;FIRST DIGIT TO R3
5478 025002 006103      ROL      R3
5479 025004 012700 000006  MOV      #6, R0      ;DIGIT COUNT
5480 025010 000404      BR       3$
5481 025012 006302      2$:     ASL      R2      ;PRINT FIRST DIGIT
5482 025014 006103      ROL      R3
5483 025016 005301      DEC      R1
5484 025020 001374      BNE      2$
5485 025022 012701 000003  3$:     MOV      #3, R1      ;DIGIT SHIFT COUNT
5486 025026 062702 000060  ADD      #50, R3      ;MAKE IT AN ASCII DIGIT
5487 025032 110367 000036  MOVB     R3, B$      ;LOAD DIGIT INTO MESSAGE
5488 025036 004567 176176  JSR      R5, $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
5489 025042 025074      .WORD   B$      ;ADDRESS OF MESSAGE TO BE TYPED
5490 025044 005003      CLR      R3      ;CLEAR INDEX
5491 025046 005300      DEC      R0      ;DEC DIGIT COUNT
5492 025050 001360      BNE      2$
5493 025052 012603      MOV      (SP)+, R3 ;POP STACK INTO R3
5494 025054 012602      MOV      (SP)+, R2 ;POP STACK INTO R2
5495 025056 012601      MOV      (SP)+, R1 ;POP STACK INTO R1
5496 025060 012600      MOV      (SP)+, R0 ;POP STACK INTO R0
5497 025062 012616      MOV      (SP)+, (SP) ;ADJUST THE STACK TO CLEAR DATA
5498 025064 004567 176150  JSR      R5, $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
5499 025070 026364      .WORD   FILL2     ;ADDRESS OF MESSAGE TO BE TYPED
5500 025072 000207      RTS      PC      ;RETURN
5501 025074 000      8$:     .BYTE   0      ;ONE DIGIT MESSAGE BUFFER
5502 025075 000      .BYTE   0      ;MESSAGE TERMINATOR

.SBTTL STANDARD PROGRAM MESSAGES
*****
;VARIOUS MESSAGE PRINTOUTS USED THRUOUT
;THE PROGRAM
*****
5509 025076 005015 052113 030461 MMAMES: .ASCIZ <15><12>'KT11 (MEMORY MANAGEMENT) AVAILABLE'
5510 025104 024040 042515 047515
5511 025112 054522 046440 047101
5512 025120 043501 046505 047105
5513 025126 024524 040440 040526
5514 025134 046111 041101 042514
5515 025142 000
5516 025143 015 046412 046505 MEMMES: .ASCIZ <15><12>'MEMORY MAP:'
5517 025150 051 17 020131 040515
5518 025156 035120 000
5519 025161 015 041012 052131 BYTMES: .ASCIZ <15><12>'BYTE MEMORY MAP:'
5520 025166 020105 042515 047515
5521 025174 054522 046440 050101
5522 025202 000072
5523 025204 005015 040520 044522 MTMAP: .ASCIZ <15><12>'PARITY MEMORY MAP:'
5524 025212 054524 046440 046505
5525 025220 051117 020131 040515
5526 025226 035120 000
5527 025231 015 043012 047522 FROM: .ASCIZ <15><12>'FROM '

```

5528	025236	020115	000		
5529	025241	040	047524	000040	TO: .ASCIZ ' TO '
5530	025246	005015	047111	052523	INSUFF: .ASCIZ <15><12>'INSUFFICIENT MEMORY...FIRST 16K NOT ALL THERE!'
5531	025254	043106	041511	042511	
5532	025262	052116	046440	046505	
5533	025270	051117	027131	027056	
5534	025276	044506	051522	020124	
5535	025304	033061	020113	047516	
5536	025312	020124	046101	020114	
5537	025320	044124	051105	020505	
5538	025326	000			
5539	025327	015	047012	020117	MTR: .ASCIZ <15><12>'NO PARITY REGISTERS FOUND'
5540	025334	040520	044522	054524	
5541	025342	051040	043505	051511	
5542	025350	042524	051522	043040	
5543	025356	052517	042116	000	
5544	025363	015	051012	051505	PWRMSG: .ASCIZ <15><12>'RESTARTING AFTER A POWER FAILURE'<15><12>
5545	025370	040524	052122	047111	
5546	025376	020107	043101	042524	
5547	025404	020122	020101	047520	
5548	025412	042527	020122	040506	
5549	025420	046111	051125	006505	
5550	025426	000012			
5551	025430	005015	047516	050040	NOPE: .ASCIZ <15><12>'NO PARITY ERRORS FOUND ON MEMORY SCAN'<15><12>
5552	025436	051101	052111	020131	
5553	025444	051105	047522	051522	
5554	025452	043040	052517	042116	
5555	025460	047440	020116	042515	
5556	025466	047515	054522	051440	
5557	025474	040503	006516	000012	
5558	025502	005015	051120	043517	PROREL: .ASCII <15><12>'PROGRAM NOW RESIDES BACK AT 0 TO 8K'
5559	025510	040522	020115	047516	
5560	025516	020127	042522	044523	
5561	025524	042504	020123	040502	
5562	025532	045503	040440	020124	
5563	025540	020060	047524	034040	
5564	025546	113			
5565	025547	015	044012	052111	.ASCIZ <15><12>'HIT CONTINUE FOR NORMAL RUNNING'<15><12>
5566	025554	041440	047117	044524	
5567	025562	052516	020105	047506	
5568	025570	020122	047516	046522	
5569	025576	046101	051040	047125	
5570	025604	044516	043516	005015	
5571	025612	000			
5572	025613	015	051012	043505	MX1: .ASCIZ <15><12>'REGISTER AT '
5573	025620	051511	042524	020122	
5574	025626	052101	000040		
5575	025632	041440	047117	051124	MX2: .ASCIZ ' CONTROLS '
5576	025640	046117	020123	000	
5577	025645	015	047012	020117	NOMEM: .ASCIZ <15><12>'NO MEMORY FOUND.'
5578	025652	042515	047515	054522	
5579	025660	043040	052517	042116	
5580	025666	000056			
5581	025670	005015	005012	047111	FADMES: .ASCII <15><12><12><12>'INPUT ALL PARAMETERS IN OCTAL.'
5582	025676	052520	020124	046101	
5583	025704	020114	040520	040522	

042524	051522
020116	041517
027114	
044506	051522
042101	051104
035123	020040
000	
000	
046012	051501
042101	051104
035123	020040
000	
000	
040477	042104
051523	044440
047125	040515
042105	041040
037513	000
051412	046105
020124	047503
040524	052116
000	
047125	054105
052103	042105
046440	051117
020131	044522
042440	051122
000	
050012	047522
046501	051040
041517	052101
052040	020117
000	
046412	051117
044124	047101
042516	050040
052111	020131
047522	020122
047125	027104
000	
051412	040503
047116	020107
042515	054522
043040	041040
042101	051101
052111	027131
000	000
050012	051101
052111	051105
020131	051105
047522	020122
046114	044527
041440	052117
000056	051101
005015	047516
046505	046440
042524	051117
000056	020131
005015	042105
044520	052123
045523	050111
043516	052040

.ASCIZ <15><12>'FIRST ADDRESS: '

LADMES: .ASCIZ <15><12>'LAST ADDRESS: '

BADADR: .ASCIZ <15><12>'ADDRESS IN UNMAPPED BANK''

CONST: .ASCIZ <15><12>'SELECT CONSTANT: '

UNEXPT: .ASCIZ <15><12>'UNEXPECTED MEMORY PARITY ERROR'

PRELOC: .ASCIZ <15><12>'PROGRAM RELOCATED TO '

MTOE: .ASCIZ <15><12>'MORE THAN ONE PARITY ERROR FOUND.'

SCANM: .ASCIZ <15><12>'SCANNING MEMORY FOR BAD PARITY.'

PEWNC: .ASCIZ <15><12>'PARITY ERROR WILL NOT CLEAR.'

NOMTST: .ASCIZ <15><12>'NO MEMORY TESTED.'

SKPMES: .ASCIZ <15><12>'SKIPPING TEST #'

5640	026356	051505	020124	000043
5641	026364	177777	000	
5642				
5643				
5644				
5645				
5646				
5647	026367	120	051101	052111
5648	026374	020131	042522	044507
5649	026402	052123	051105	042040
5650	026410	052101	020101	051105
5651	026416	047522	027122	000
5652	026423	101	042104	042522
5653	026430	051523	052040	051505
5654	026436	020124	051105	047522
5655	026444	024122	051524	030524
5656	026452	032455	027051	000
5657	026457	103	047117	052123
5658	026464	047101	020124	040504
5659	026472	040524	042440	051122
5660	026500	051117	052050	052123
5661	026506	026466	030061	027051
5662	026514	000		
5663	026515	122	052117	052101
5664	026522	047111	020107	044502
5665	026530	020124	051105	047522
5666	026536	024122	051524	030524
5667	026544	026461	031061	027051
5668	026552	000		
5669	026553	061	054040	051117
5670	026560	034040	050040	052101
5671	026566	042524	047122	042440
5672	026574	051122	051117	052050
5673	026602	052123	031461	027051
5674	026610	000		
5675	026611	063	054040	051117
5676	026616	034440	050040	052101
5677	026624	042524	047122	042440
5678	026632	051122	051117	052050
5679	026640	052123	032061	030455
5680	026646	024467	000056	
5681	026652	020070	047530	020122
5682	026660	031461	050040	052101
5683	026666	042524	047122	042440
5684	026674	051122	051117	052050
5685	026702	052123	030062	027051
5686	026710	000		
5687	026711	120	051101	052111
5688	026716	020131	042515	047515
5689	026724	054522	040440	042104
5690	026732	042522	051523	042440
5691	026740	051122	051117	052050
5692	026746	052123	030462	027051
5693	026754	000		
5694	026755	104	052101	050111
5695	026762	053440	052111	020110

FILL2: .ASCIZ (377)(377)

.SBTTL ERROR REPORTING MESSAGES AND TABLES.

* MESSAGE BLOCK FOR ERROR TABLE TYPEOUTS

DM1: .ASCIZ 'PARITY REGISTER DATA ERROR.'

DM2: .ASCIZ 'ADDRESS TEST ERROR(TST1-5).'

DM4: .ASCIZ 'CONSTANT DATA ERROR(TST6-10).'

DM5: .ASCIZ 'ROTATING BIT ERROR(TST11-12).'

DM6: .ASCIZ '1 XOR 8 PATTERN ERROR(TST13).'

DM7: .ASCIZ '3 XOR 9 PATTERN ERROR(TST14-17).'

DM10: .ASCIZ '8 XOR 13 PATTERN ERROR(TST20).'

DM11: .ASCIZ 'PARITY MEMORY ADDRESS ERROR(TST21).'

DM12: .ASCIZ 'DATIP WITH WRONG PARITY DIDN'T TRAP(TST21).'

5696	026770	051127	047117	020107
5697	026776	040520	044522	054524
5698	027004	042040	042111	023516
5699	027012	020124	051124	050101
5700	027020	052050	052123	030462
5701	027026	027051	000	
5702	027031	127	047522	043516
5703	027036	050040	051101	052111
5704	027044	020131	051124	050101
5705	027052	042520	026104	041040
5706	027060	052125	047040	020117
5707	027066	042522	044507	052123
5708	027074	051105	051440	047510
5709	027102	051527	042440	051122
5710	027110	051117	043040	040514
5711	027116	027107	000	
5712	027121	120	051101	052111
5713	027126	020131	042522	044507
5714	027134	052123	051105	047040
5715	027142	052117	046440	050101
5716	027150	042520	020104	051501
5717	027156	041440	047117	051124
5718	027164	046117	044514	043516
5719	027172	052040	044510	020123
5720	027200	042101	051104	051505
5721	027206	044123	051524	031124
5722	027214	024461	000056	
5723	027220	047515	042522	052040
5724	027226	040510	020116	047117
5725	027234	020105	042522	044507
5726	027242	052123	051105	044440
5727	027250	042116	041511	052101
5728	027256	042105	050040	051101
5729	027264	052111	020131	051105
5730	027272	047522	027122	000
5731	027277	104	052101	020101
5732	027304	044123	052517	042114
5733	027312	023516	020124	040510
5734	027320	042526	041440	040510
5735	027326	043516	042105	053440
5736	027334	042510	020116	040520
5737	027342	044522	054524	042440
5738	027350	051122	051117	052040
5739	027356	040522	050120	042105
5740	027364	052050	052123	030462
5741	027372	027051	000	
5742	027375	122	047101	047504
5743	027402	020115	040504	040524
5744	027410	042440	051122	051117
5745	027416	052050	052123	031062
5746	027424	027051	000	
5747	027427	111	051516	051124
5748	027434	041525	044524	047117
5749	027442	042440	042530	052503
5750	027450	044524	047117	042440
5751	027456	051122	051117	052050

DM13: .ASCIZ 'WRONG PARITY TRAPPED, BUT NO REGISTER SHOWS ERROR FLAG.'

DM14: .ASCIZ 'PARITY REGISTER NOT MAPPED AS CONTROLLING THIS ADDRESS(TST2!).'

DM16: .ASCIZ 'MORE THAN ONE REGISTER INDICATED PARITY ERROR.'

DM17: .ASCIZ 'DATA SHOULDN'T HAVE CHANGED WHEN PARITY ERROR TRAPPED(TST2!).'

DM20: .ASCIZ 'RANDOM DATA ERROR(TST22).'

DM21: .ASCIZ 'INSTRUCTION EXECUTION ERROR(TST23-30).'

5752	027464	052123	031462	031455					
5753	027472	024460	000056						
5754	027476	041042	040522	041516	DM22:	.ASCIZ	'"	BRANCH GOBBLE"	ERROR(TST31).'
5755	027504	020110	047507	041102					
5756	027512	042514	020042	051105					
5757	027520	047522	024122	051524					
5758	027526	031524	024461	000056					
5759	027534	051120	043517	040522	DM23:	.ASCIZ	'PROGRAM	CODE CHANGED	WHEN RELOCATED.'
5760	027542	020115	047503	042504					
5761	027550	041440	040510	043516					
5762	027556	042105	053440	042510					
5763	027564	020116	042522	047514					
5764	027572	040503	042524	027104					
5765	027600	000							
5766	027601	124	040522	050120	DM24:	.ASCIZ	'TRAPPED,	BUT NO REGISTER	HAD ERROR BIT SET.'
5767	027606	042105	020054	052502					
5768	027614	020124	047516	051040					
5769	027622	043505	051511	042524					
5770	027630	020122	040510	020104					
5771	027636	051105	047522	020122					
5772	027644	044502	020124	042523					
5773	027652	027124	000						
5774	027655	124	040522	050120	DM25:	.ASCIZ	'TRAPPED	TO 114.'	
5775	027662	042105	052040	020117					
5776	027670	030461	027064	000					
5777	027675	106	044501	042514	DM26:	.ASCIZ	'FAILED	TO TRAP.'	
5778	027702	020104	047524	052040					
5779	027710	040522	027120	000					
5780	027715	050	041501	044524	DM27:	.ASCIZ	'"	(ACTION ENABLE	WASN'T SET)."
5781	027722	047117	042440	040516					
5782	027730	046102	020105	040527					
5783	027736	047123	052047	051440					
5784	027744	052105	027051	000					
5785	027751	015	052012	040522	DM31:	.ASCIZ	'<15><12>	'TRAPPED	TO 4 '
5786	027756	050120	042105	052040					
5787	027764	020117	020064	000					
5788									
5789									
5790									
5791									
5792									
5793	027771	120	004503	042522	DM1:	.ASCIZ	'PC	REG	S/B WAS'
5794	027776	004507	027523	004502					
5795	030004	040527	000123						
5796	030010	027526	041520	050011	DM2:	.ASCIZ	'V/PC	P/PC	MA S/B WAS'
5797	030016	050057	004503	040515					
5798	030024	051411	041057	053411					
5799	030032	051501	000						
5800	030035	126	050057	004503	DM12:	.ASCIZ	'V·PC	P/PC	MA S/B'
5801	030042	027520	041520	046411					
5802	030050	004501	027523	000102					
5803	030056	027526	041520	050011	DM14:	.ASCIZ	'V/PC	P/PC	REG MA'
5804	030064	050057	004503	042522					
5805	030072	004507	040515	000					
5806	030077	126	050057	004503	DM15:	.ASCIZ	'V·PC	P·PC	MAUT REG S B WAS'
5807	030104	027520	041520	046411					

```

;*****
;DATA COLUMN HEADINGS
;*****

```

F10

5808	030112	052501	004524	042522																
5809	030120	004507	027523	004502																
5810	030126	040527	000123																	
5811	030132	027526	041520	050011	DH21:	.ASCIZ	'V/PC	P/PC	IUT	MA	S/B	WAS'								
5812	030140	050057	004503	052511																
5813	030146	004524	040515	051411																
5814	030154	041057	053411	051501																
5815	030162	000																		
5816	030163	126	050057	004503	DH22:	.ASCIZ	'V/PC	P/PC	PS S/B	PS WAS	MA	S/B	WAS'							
5817	030170	027520	041520	050011																
5818	030176	020123	027523	004502																
5819	030204	051520	053440	051501																
5820	030212	046411	004501	027523																
5821	030220	004502	040527	000123																
5822	030226	027526	041520	050011	DH23:	.ASCIZ	'V/PC	P/PC	SRC MA	DST MA	S/B	WAS'								
5823	030234	050057	004503	051123																
5824	030242	020103	040515	042011																
5825	030250	052123	046440	004501																
5826	030256	027523	004502	040527																
5827	030264	000123																		
5828	030266	027526	041520	050011	DH24:	.ASCIZ	'V/PC	P/PC	TRP/PC'											
5829	030274	050057	004503	051124																
5830	030302	027520	041520	000																
5831	030307	126	050057	004503	DH25:	.ASCIZ	'V/PC	P/PC	TRP/PC	REG	WAS'									
5832	030314	027520	041520	052011																
5833	030322	050122	050057	004503																
5834	030330	042522	004507	040527																
5835	030336	000123																		
5836	030340	027526	041520	050011	DH26:	.ASCIZ	'V/PC	P/PC	REG	WAS'										
5837	030346	050057	004503	042522																
5838	030354	004507	040527	000123																
5839	030362	042522	004507	040527	DH30:	.ASCIZ	'REG	WAS	MA	WAS'										
5840	030370	004523	040515	053411																
5841	030376	051501	000																	

```

:*****
:* DATA FORMAT TABLE FOR ERROR PRINTOUT.
:*****

```

5846	030401	000	377	000	DF1:	.BYTE	0,-1,0,0
5847	030404	000					
5848	030405	000	377	377	DF2:	.BYTE	0,-1,-1,0,0
5849	030410	000	000				
5850	030412	000	377	377	DF3:	.BYTE	0,-1,-1,-2,-2
5851	030415	376	376				
5852	030417	000	377	377	DF14:	.BYTE	0,-1,-1,-1,0,0
5853	030422	377	000	000			
5854	030425	000	377	000	DF21:	.BYTE	0,-1,0,-1,0,0
5855	030430	377	000	000			
5856	030433	000	377	376	DF22:	.BYTE	0,-1,-2,-2,-1,0,0
5857	030436	376	377	000			
5858	030441	000					
5859	030442	377	000	377	DF30:	.BYTE	-1,0,-1,-2
5860	030445	376					
5861						.EVEN	

032110

. = 32110

:THE LOADERS ARE SAVE HERE TO END OF BK

G10

MAINDEC-11-DZQMC-B-D: 0-124K MEMORY EXERCISER, 16K VER
DZQMCB.P11 .. ERROR REPORTING MESSAGES AND TABLES.

MACY11 27(732) 10-SEP-76 12:01 PAGE 124

5964
5965

000001

.END

MAINDEC-11-DZQMC-B-D: 0-124K MEMORY EXERCISER, 16K VER
DZQMCB.P11 CROSS REFERENCE TABLE -- USER SYMBOLS

RBASE = 000000	968	1015		
ACDW1 = 000000	968	1017		
ACDW2 = 000000	968	1019		
ACPUOP = 000000	968	986		
ACDWO = 000000	968	1019		
ACDW1 = 000000	968	1020		
ACDW10 = 000000	968	1029		
ACDW11 = 000000	968	1030		
ACDW12 = 000000	968	1031		
ACDW13 = 000000	968	1032		
ACDW14 = 000000	968	1033		
ACDW15 = 000000	968	1034		
ADDW2 = 000000	958	1021		
ADDW3 = 000000	968	1022		
ADDW4 = 000000	968	1023		
ADDW5 = 000000	968	1024		
ADDW6 = 000000	968	1025		
ADDW7 = 000000	968	1026		
ADDW8 = 000000	968	1027		
ADDW9 = 000000	968	1028		
ADEVCT = 000000	968	977		
ADEVN = 000000	968	1016		
AE = 000001	768*	3177	3262	4409
AENV = 000000	968	982		
AENVN = 000000	968	993		
AFATAL = 000000	968	974		
AMADR1 = 000000	968	999		
AMADR2 = 000000	968	1003		
AMADR3 = 000000	968	1006		
AMADR4 = 000000	968	1009		
AMAMS1 = 000000	968	993		
AMAMS2 = 000000	968	1001		
AMAMS3 = 000000	968	1004		
AMAMS4 = 000000	968	1007		
AMSGAD = 000000	968	979		
AMSGLG = 000000	968	980		
AMSGTY = 000000	968	973		
AMTYP1 = 000000	968	994		
AMTYP2 = 000000	968	1002		
AMTYP3 = 000000	968	1005		
AMTYP4 = 000000	968	1008		
APASS = 000000	968	976		
APRIOR = 000000	968	1012		
APTCSU = 000040	5140	5284*		
APTENV = 000001	4836	5133	5203	5282*
APTSIZ = 000200	1447	5281*		
APTSPO = 000100	5135	5205	5283*	
ASWREG = 000000	968	984		
AYESTN = 000000	968	975		
AUNIT = 000000	968	978		
AJSUR = 000000	968	985		
AVECT1 = 000000	968	1010		
AVECT2 = 000000	968	1011		
BACADR C26000	1927	5595*		
BANKNO 016432	2114	2123	2150	2160 4075*
BGENO 014174	3665	3691	3693*	

N10

MAINDEC-11-DZQMC-B-D: 0-124K MEMORY EXERCISER, 16K VER
DZQMCB.P11 CROSS REFERENCE TABLE -- USER SYMBOLS

MACY11 27(732) 10-SEP-76 12:01 PAGE 132

RANTST	013074	3297*												
RELOC	016640	4159*	4229	4258	4295	4305								
RELOC F	000600	857	876*	1466*	1965	3911	4261*	4265*	4274	4282	4303	4327*	4399	4722
		4878	4899	4901	5101	5309								
RELTOP	016762	3742	4195*											
RELO	017364	864	3745	4288*										
RESCHK	004230	1722*												
RESLDR	017572	870	3746	4337*										
RESRVD	001514	1095*	1147	1687	1692	1697	1707	3230	3231	3260				
RESTAR	000300	789	814*	913	1463									
RESTOR	000304	790	816*											
REST1	000306	815	817*											
REST2	000324	819	821*											
RESVEC=	000010	714*												
ROTATE	016512	2268	2292	4104*										
RW	= 000006	764*	3803	3804	3805	3806	3810							
RO	=%000000	635*	825*	827*	830*	831*	832*	833*	834*	835*	836*	837*	838*	840*
		846*	849	851*	855	857*	858*	859	860*	861	886	907*	1460*	1461
		1471*	1472	1474*	1478	1480	1508*	1510*	1523*	1563	1565*	1566*	1567	1590*
		1598	1608	1617*	1630	1684*	1692	1694	1700	1712*	1830*	1831	1837*	1840*
		1850*	1871*	1875	1902	1904	1913*	1916	1921	1930	1992	1994	2000*	2009*
		2011	2036	2037*	2046*	2048	2074*	2075*	2076	2085*	2088	2094*	2115	2125
		2151*	2152	2161*	2163	2191*	2193	2206*	2210	2242*	2245	2247	2264*	2271
		2288*	2295	2314*	2317*	2320	2322	2325	2327	2330	2332	2335	2337	2349*
		2354	2360*	2362	2368*	2370	2376*	2378	2384*	2387*	2410*	2415	2421*	2423
		2429*	2431	2437*	2439	2445*	2448*	2462*	2473*	2478	2485	2492	2499	2505*
		2508*	2516*	2522	2528*	2531	2537*	2540	2548*	2551*	2565*	2577*	2582	2589
		2596	2603	2609*	2612*	2620*	2626	2632*	2635	2641*	2644	2652*	2655*	2669*
		2680*	2686	2693	2700	2707	2713	2714*	2718	2719*	2728*	2734	2740*	2743
		2749*	2752	2759	2765*	2768	2774*	2777	2784	2790*	2793	2799*	2802	2809
		2815*	2818	2824*	2827	2833	2834*	2838	2839*	2855*	2866*	2872	2879	2896
		2893	2899	2900*	2904	2905*	2914*	2920	2926*	2929	2935*	2938	2945	2951*
		2954	2962*	2963	2970	2976*	2979	2985*	2988	2995	3001*	3004	3010*	3013
		3019	3020*	3024	3025*	3040*	3043*	3045*	3046	3058*	3061*	3063*	3066	3073
		3106*	3109*	3111*	3114	3121	3151*	3174*	3175	3181	3219	3220*	3221*	3222*
		3223*	3224*	3227*	3228*	3229*	3230*	3233	3262*	3263	3271*	3272	3278	3280*
		3310*	3312	3360*	3366	3410*	3416	3460*	3467	3511*	3517	3561*	3567	3611*
		3618	3650*	3652	3667*	3686	3692	3780*	3783	4054*	4058*	4075*	4093*	4096
		4129	4130	4131	4132	4139	4140	4141	4142	4151	4152*	4199	4204*	4208*
		4212	4215	4219*	4223	4226*	4227	4238*	4241*	4243	4246*	4248	4250	4251*
		4252*	4253	4279	4281*	4311	4312*	4313*	4314	4316	4318*	4337*	4345*	4449
		4459*	4471	4510*	4515*	4528*	4569	4577	4579	4594	4596	4865	4866*	4867*
		4879*	4880*	4881*	4882*	4883*	4884	4890	4897	4900*	4901*	4902	4917	4939*
		5052	5060*	5061	5064	5084*	5087*	5131	5132*	5137	5142	5145*	5199	5207*
		5211	5212	5214*	5215*	5216	5227*	5232	5238	5253*	5257	5258*	5259*	5262
		5265*	5268	5275*	5299	5309*	5311	5314	5330	5331	5344*	5451	5470*	5473*
		5479*	5491*	5496*										
R1	=%000001	636*	826*	828*	841*	844*	853*	887	906*	1472*	1480*	1509*	1511*	1524*
		1564	1686*	1687*	1695*	1697*	1700	1706*	1707*	1729*	1731*	1732	1838*	1839*
		1851*	1872*	1877	1903	1905	1914*	1918	1923	1931	1993*	1994	2010*	2011
		2047*	2048	2087*	2088	2124*	2125	2162*	2163	2209*	2210	2246*	2247	2269*
		2271	2293*	2295	2353*	2354	2361*	2362	2369*	2370	2377*	2378	2414*	2415
		2422*	2423	2430*	2431	2439*	2439	2477*	2478	2484*	2485	2492	2498*	2498*
		2499	2521*	2522	2530*	2531	2539*	2540	2581*	2592	2588*	2589	2595*	2596
		2602*	2603	2625*	2626	2634*	2635	2643*	2644	2685*	2686	2692*	2693	2699*
		2700	2706*	2707	2733*	2734	2742*	2743	2751*	2752	2758*	2759	2767*	2768

3598	3703	3704	4134	4135	4136	4137	4144	4145	4146	4147	4152	4153*
4161	4164*	4166*	4170	4173	4175	4186	4189*	4363	4364*	4365	4367	4373
4373	4387*	4407*	4408*	4409*	4410	4412*	4422	4423*	4424	4426	4438	4440*
4452	4479*	4479	4490*	4492	4497*	4498	4525*	4535	4536*	4537	4539*	4542*
4554	4555	4558	4566	4588	4592*	4593	4598	4600*	4605	4607*	4612	4614*
4616*	4625	4634*	4635*	4641*	4968	4970*	4971	4977*	4978	4986*	4987	4989
4999	5004	5006*	5016	5019	5021*	5025*	5302	5310*	5311*	5312*	5326*	5329*
5339*	5339*	5341*	5389	5398*	5404*	5405*	5408*	5413*	5414*	5415	5425*	5454
5456*	5472*	5476*	5478*	5482*	5486*	5487	5490*	5493*				
6329*	890	903*	1514*	1524	1553*	1648*	1656*	1665*	1672*	1879*	1895	1990*
2007*	2034*	2044*	2072*	2083*	2113*	2122*	2149*	2159*	2192*	2207*	2240*	2242*
2244*	2266*	2290*	2316*	2319*	2340*	2350*	2351*	2385*	2395*	2396*	2401*	2411*
2412*	2446*	2464*	2474*	2475*	2506*	2517*	2518*	2549*	2567*	2578*	2579*	2610*
2621*	2622*	2653*	2671*	2682*	2683*	2716*	2730*	2731*	2836*	2857*	2868*	2869*
2907*	2916*	2917*	3022*	3041*	3042*	3049*	3059*	3060*	3081*	3092*	3093*	3095*
3107*	3109*	3129*	3153*	3252*	3270*	3299*	3359*	3360	3361*	3363	3409*	3411*
3413	3459*	3461*	3463	3510*	3512*	3514	3560*	3562*	3564	3610*	3612*	3614
3647*	3675*	3676*	3681*	3688*	3919	3920*	4095*	4127*	4149*	4162	4165*	4167*
4169*	4181*	4188*	4453	4462*	4481	4484*	4517	4524*	4599	4593*	4601*	4608*
4612	4615*	4617*	4623*	4636*	4640*	4662*	4748	4750	4752	4758	4750	5330
5392*	5393*	5394*	5395	5396*	5410	5412*	5421*	5424*				
6408	814*	816*	865	891	902*	910*	1455*	1493*	1516*	1530*	1532*	1533*
1524*	1535*	1536*	1538*	1540	1561*	1569*	1587*	1600*	1610*	1632*	1669*	1812*
1819*	1827*	1854*	1881*	1926*	1938*	1933*	2001	2017	2030*	2038	2054	2069*
2022*	2095	2109*	2116	2131	2145*	2153	2169	2198*	2203*	2203	2216	2227*
2252	2251*	2277*	2285*	2301	2308*	2342	2388	2403	2449	2456*	2466	2509
2552*	2553*	2569	2613	2656	2663*	2673	2721	2841	2849*	2859	2907	3027*
3024*	3051	3083	3098	3131	3143*	3158	3265	3294*	3306	3322	3352*	3354*
3373	3402*	3414*	3423	3452*	3464*	3474	3503*	3515*	3524	3553*	3565*	3574*
3603*	3615*	3625	3640*	3653	3686*	3692*	3697	3700	3713*	3723	3728*	3747*
3769*	3776*	3829*	3840*	3852	3866*	3879	3888*	3890*	3900*	3900*	3948*	3952
3967*	3987*	3989	3999*	4001*	4046*	4097	4163	4164	4179*	4183*	4191*	4229*
4258*	4295*	4305*	4359*	4377*	4456*	4494*	4519*	4581*	4602*	4609*	4662	4663
4712	4732*	4776*	4791*	4792	4793*	4796*	4797	4798*	4824*	4833*	4863*	4886*
4888*	4892*	4894*	4933*	4939*	4983*	4990*	4996*	5001*	5027*	5009*	5013*	5017*
5022*	5088*	5090*	5100	5107*	5153*	5303	5305*	5307*	5315*	5319*	5334	5340*
5345*	5391	5397*	5399*	5401*	5402*	5423*	5404	5416*	5423*	5488*	5498*	
6418	643	1429*	1430*	1431								
642*	644											
1705*	1563*	1564*	1862*	1863*	1904*	1935*	3732	3734	4739	4739		
4457	5622*											
787	1427*											
1121*	1425*	1427*	1842									
3161	4400	4402	4405*									
2265	2289	3152	4094*									
4777	5638*											
643*	817*	853*	866*	887*	888*	889*	890*	891*	892	898*	902	903
904	905	906	907	912*	1433*	1436*	1437*	1443	1444	1445	1475	1479*
1483	1507*	1545	1579*	1603*	1607	1614*	1636*	1559	1822*	1825*	1859*	1861*
1869	1986*	1888	1892*	1894	1898	1920	1925	1934*	1935*	1936	1943*	1945
1963*	2224*	2225*	2226	2713*	2715	2718*	2720	2833*	2835	2838*	2840	2899*
2901	2904*	2906	3019*	3021	3024*	3026	3090*	3101	3198	3219*	3240*	3258
3271	3300*	3301*	3308	3309	3648*	3649*	3666	3716	3720*	3722	3731*	3735*
4004*	4047*	4058*	4069	4076*	4077*	4087	4098	4151*	4153	4160*	4161*	4162*
4186*	4188	4189	4190	4199*	4200*	4215*	4216*	4225	4226	4252*	4253	4254*
4280	4281	4282*	4303*	4304	4309	4310*	4311*	4313	4319	4319	4320	4321

74 =:0000004

75 =:0000005

76 =:0000006

77 =:0000007

USVYST 0001532

USCANM 0028211

USLECT 0028222

USMFLG 0019554

USMTRM 0200020

USMTCN 0164722

USMTRM 0262422

78 =:0000008

J11

MAINDEC-11-DZQMC-B-D: 0-124K MEMORY EXERCISER, 16K VER
DZQMCB.P11 CROSS REFERENCE TABLE -- USER SYMBOLS

MACY11 27(732) 10-SEP-76 12:01 PAGE 141

.5X	=	001326	1042#	1047
.3X9		007324	2462#	
.9X13		011674	3040#	

12

ABORT	592#	848	3842	3875	3913	4085	4178	4197	4255	4290	4339				
CKWD	592#	1700	1994	2011	2088	2125	2163	2210	2247	2271	2295	2354	2362	2370	2378
	2415#	2423	2431	2439	2478	2485	2492	2499	2522	2531	2540	2582	2589	2596	2603
	2626	2635	2644	2686	2693	2700	2707	2734	2743	2752	2759	2768	2777	2784	2793
	2822	2809	2818	2827	2872	2879	2886	2893	2920	2929	2938	2945	2954	2963	2970
	2979	2988	2995	3004	3013	3066	3073	3114	3121	3233	3263	3312	3366	3416	3467
	3517	3567	3618												
CKWD2	592#	1993	2086	2208	2246	2352	2361	2369	2377	2413	2422	2430	2438	2476	2484
	2491	2498	2520	2530	2539	2580	2588	2595	2602	2624	2634	2643	2684	2692	2699
	2706	2732	2742	2751	2758	2767	2776	2783	2792	2801	2808	2817	2826	2870	2878
	2885	2892	2918	2928	2937	2944	2953	2962	2969	2978	2987	2994	3003	3012	3064
	3072	3112	3120	3311											
COMMEN	1#	592#	725#	1419	1957										
ENDCOM	1#	592#	603	725#	1423	1961									
ERROR	625#														
ESCAPE	1#	725#													
GETPRI	1#														
LDPDR	592#	3802	3904	3805	3806	3810									
MORETA	918#	1037													
MULT	1#	725#													
NEWTST	1#	592#	725#	1973	2020	2058	2099	2135	2177	2198	2232	2257	2281	2304	2452
	2555	2659	2845	3030	3134	3290	3326	3376	3426	3477	3527	3577	3628		
POP	1#	592#	725#	902	3258	3271	4069	4086	4188	4280	4318	4386	4412	4440	4503
	4513	4521	4541	4633	4640	5082	5252	5253	5265	5274	5340	5493			
PRINT	592#	1493	1516	1561	1569	1586	1600	1610	1632	1669	1813	1819	1827	1854	1880
	1926	1937	3747	4183	4359	4377	4456	4494	4519	4581	4732	4775	5488	5498	
PUSH	1#	592#	725#	886	3218	3240	4058	4076	4159	4198	4215	4311	4362	4407	4422
	4448	4468	4475	4534	4585	4611	5052	5198	5200	5221	5256	5267	5298	5450	
RDCHR	592#	4973													
RDDEC	592#														
RDLIN	592#	5055													
RDOCT	592#	1957	1884	1941											
REPORT	1#	592#													
RESREG	592#														
SAVREG	592#														
SCOPE	626#														
SCOPEX	4646#	4712													
SCOPIN	4646#	4662													
SETPRI	1#														
SETUP	1#	592#	725#	1428											
SIMTRP	592#	1823	1857	1884	1941	3772	4619	4627	4781	4872	4905	4912	4925	4973	5056
	5102														
SKIP	1#	725#	2243												
SLASH	1#	592#	725#	1213	1223										
SPACE	725#														
STARS	1#	592#	725#	797	808	813	879	918	964	965	1037	1041	1048	1059	1091
	1093	1140	1144	1159	1161	1194	1196	1497	1506	1574	1577	1643	1646	1676	1680
	1717	1720	1741	1750	1761	1765	1807	1810	1847	1849	1973	1981	2020	2028	2058
	2066	2099	2107	2135	2143	2173	2176	2177	2186	2198	2201	2232	2235	2257	2259
	2281	2283	2304	2306	2346	2348	2392	2394	2407	2409	2452	2454	2470	2472	2513
	2515	2555	2557	2574	2576	2617	2619	2659	2661	2677	2679	2725	2727	2845	2847
	2863	2865	2911	2913	3030	3032	3055	3057	3087	3089	3103	3105	3134	3141	3290
	3292	3326	3350	3376	3400	3426	3450	3477	3501	3527	3551	3577	3601	3628	3639
	3749	3795	3801	3823	3825	3922	3927	4012	4017	4050	4053	4072	4074	4091	4093
	4101	4103	4124	4126	4156	4158	4192	4194	4285	4287	4332	4336	4353	4357	4391
	4394	4415	4417	4443	4447	4531	4533	4546	4549	4573	4576	4646	4802	4854	4944

EXERCISER 16K VER
-- PERMANENT SYMBOLS

REFERENCE TABLE

MEM	MEM	MEM	MEM	MEM	MEM	MEM	MEM	MEM	MEM	MEM	MEM	MEM	MEM	MEM
1482	1489	1522	1581	1595	1599	1621	1482	1489	1522	1581	1595	1599	1621	1621
1799	1802	1919	1941	1942	1976	1989	1799	1802	1919	1941	1942	1976	1989	1989
2018	2039	2055	2078	2099	2117	2132	2018	2039	2055	2078	2099	2117	2132	2132
2341	2343	2386	2389	2426	2404	2447	2341	2343	2386	2389	2426	2404	2447	2447
2611	2614	2651	2654	2657	2674	2717	2611	2614	2651	2654	2657	2674	2717	2717
3048	3050	3052	3090	3092	3084	3097	3048	3050	3052	3090	3092	3084	3097	3097
3185	3201	3212	3214	3245	3204	3207	3185	3201	3212	3214	3245	3204	3207	3207
3641	3664	3721	3733	3737	3739	3741	3641	3664	3721	3733	3737	3739	3741	3741
4182	4211	4222	4228	4278	4278	4279	4182	4211	4222	4228	4278	4278	4279	4279
4766	4782	4799	4828	4839	4831	4838	4766	4782	4799	4828	4839	4831	4838	4838
5141	5141	5143	5151	5159	5172	5179	5141	5141	5143	5151	5159	5172	5179	5179
5492	5492	5494	5494	5499	5480	5483	5492	5492	5494	5494	5499	5480	5483	5483
5874	5874	5874	5874	5884	5865	5868	5874	5874	5874	5874	5884	5865	5868	5868
6084	6084	6084	6084	6089	6080	6083	6084	6084	6084	6084	6089	6080	6083	6083
6376	6376	6376	6376	6389	6380	6383	6376	6376	6376	6376	6389	6380	6383	6383
6425	6425	6425	6425	6440	6430	6433	6425	6425	6425	6425	6440	6430	6433	6433
6480	6480	6480	6480	6498	6489	6492	6480	6480	6480	6480	6498	6489	6492	6492
6493	6493	6493	6493	6500	6490	6493	6493	6493	6493	6493	6500	6490	6493	6493
6800	6800	6800	6800	6814	6805	6808	6800	6800	6800	6800	6814	6805	6808	6808
7179	7179	7179	7179	7199	7190	7193	7179	7179	7179	7179	7199	7190	7193	7193

SUB	2009 4240 5214	2094 4245 5259	2228 4308 5315	2229 4310	3305 4313	3320 4319	3661 4320	3698 4321	3701 4324	3979 4325	4027 4326	4037 4568	4179 4600	4206 4828	4217 4878
SWAB TST	2222 918 1732 3184 4291 4479 4771 5224 5227 5227	821 1738 3225 4314 4481 4773 5242 5336	865 1756 3657 4340 4492 4843 5244 3279	1439 1775 3717 4349 4498 4849 5260 4491	1469 1777 3718 4365 4506 4931 5270 4598	1527 1790 3722 4367 4517 4980 5320 4605	1548 1831 3740 4373 4537 4993 5330 4694	1598 1888 3820 4375 4577 5024 5410 4902	1608 1990 3902 4395 4579 5079 5435 4956	1622 1925 3933 4399 4671 5096 5457 5127	1630 1922 4007 4410 4697 5144	1554 1546 4025 4418 4716 5152	1667 1967 4056 4424 4722 5173	1714 3146 4201 4438 4742 5209	1724 3179 4275 4466 4749 5230
.ASC11 .ASC12	961 960 5551 5638 5742 5816 5033 1273 928 1005 4631 5431 586 3596 1017 1197 175 1989 2040 2093 2136 2170 2255 2303 2374 2427 2471 2527 2577 2639 2690 2748 2813 2864 2925 2986	962 963 5565 5641 5747 5822	5558 1459 5572 5647 5754 5828	5581 3789 5575 5652 5759 5831	4942 5577 5657 5766 5836	5032 5587 5663 5774 5839	5509 5591 5669 5777	5516 5595 5675 5780	5519 5600 5681 5785	5523 5604 5687 5793	5527 5610 5694 5796	5529 5615 5702 5800	5530 5622 5712 5803	5539 5628 5723 5806	5544 5634 5731 5811
.BLKB .BLKM .BYTM	5033 1273 928 1005 4631 5431 586 3596 1017 1197 175 1989 2040 2093 2136 2170 2255 2303 2374 2427 2471 2527 2577 2639 2690 2748 2813 2864 2925 2986	5354 929 1007 4632 5432 592	934 1008 4785 5501	935 1010 4786 5502	950 1011 4840 5846	951 1012 4841 5848	952 1013 4929 5850	953 1121 4930 5852	982 1122 5030 5854	983 1123 5031 5856	993 3683 5277 5859	994 3684 5279	1001 3792 5279	1002 4623 5429	1004 4624 5430
.FNAB1 .FNAB2 .FNAB3	586 3596 1017 1197 175 1989 2040 2093 2136 2170 2255 2303 2374 2427 2471 2527 2577 2639 2690 2748 2813 2864 2925 2986	599 804 959 1018 1033 1214 1644 1751 1998 2041 2097 2137 2180 2218 2259 2303 2374 2428 2473 2527 2577 2640 2691 2756 2814 2866 2927 2988	603 806 960 1019 1034 1224 1647 1762 1999 2052 2098 2143 2186 2231 2263 2305 2382 2436 2483 2544 2593 2648 2697 2757 2822 2876 2934 2999	613 909 961 1020 1037 1276 1677 1766 2023 2053 2100 2144 2187 2231 2263 2305 2382 2436 2483 2544 2593 2649 2698 2763 2823 2877 2942 3000	615 814 965 1021 1038 1421 1681 1808 2004 2056 2101 2147 2190 2233 2264 2306 2383 2443 2490 2545 2594 2658 2704 2764 2831 2883 2949 3008	616 880 966 1022 1042 1424 1691 1811 2015 2059 2108 2155 2196 2234 2275 2313 2390 2444 2496 2555 2601 2659 2711 2772 2843 2894 2950 3017	617 892 993 1023 1049 1433 1692 1849 2016 2060 2111 2156 2199 2239 2280 2314 2393 2451 2497 2556 2607 2661 2712 2773 2844 2897 2958 3029	625 902 1001 1024 1060 1434 1707 1850 2019 2060 2111 2156 2199 2239 2280 2314 2393 2451 2497 2556 2607 2661 2712 2773 2844 2897 2958 3029	711 912 1004 1025 1092 1436 1705 1959 2020 2066 2112 2156 2199 2239 2280 2314 2393 2451 2497 2556 2607 2661 2712 2773 2844 2897 2958 3029	725 914 1007 1026 1094 1451 1711 1962 2021 2067 2118 2168 2200 2240 2282 2314 2393 2451 2497 2556 2607 2661 2712 2773 2844 2897 2958 3029	740 918 1010 1027 1141 1455 1712 1974 2022 2070 2119 2171 2202 2244 2284 2314 2393 2451 2497 2556 2607 2661 2712 2773 2844 2897 2958 3029	751 919 1011 1028 1145 1459 1718 1975 2028 2071 2129 2172 2202 2244 2284 2314 2393 2451 2497 2556 2607 2661 2712 2773 2844 2897 2958 3029	762 926 1012 1029 1160 1499 1721 1981 2029 2079 2130 2174 2202 2244 2284 2314 2393 2451 2497 2556 2607 2661 2712 2773 2844 2897 2958 3029	774 928 1013 1030 1162 1507 1736 1982 2032 2080 2133 2177 2202 2244 2284 2314 2393 2451 2497 2556 2607 2661 2712 2773 2844 2897 2958 3029	797 954 1016 1031 1195 1575 1737 1988 2033 2092 2134 2178 2202 2244 2284 2314 2393 2451 2497 2556 2607 2661 2712 2773 2844 2897 2958 3029

3106	3118	3119	3125	3126	3133	3134	3135	3136	3141	3142	3145	3146	3171	3172
3193	3194	3209	3210	3217	3218	3237	3238	3250	3251	3267	3268	3276	3277	3291
3292	3293	3296	3297	3316	3317	3324	3325	3327	3328	3350	3351	3357	3358	3370
3371	3375	3376	3377	3378	3400	3401	3407	3408	3420	3421	3425	3426	3427	3428
3450	3451	3457	3458	3471	3472	3476	3477	3478	3479	3501	3502	3508	3509	3521
3522	3526	3527	3528	3529	3551	3552	3558	3559	3571	3572	3576	3577	3578	3579
3601	3602	3608	3609	3622	3623	3627	3628	3629	3630	3638	3639	3645	3646	3712
3713	3730	3750	3753	3754	3755	3757	3760	3765	3768	3769	3778	3779	3788	3789
3792	3793	3796	3802	3824	3826	3923	3928	4013	4018	4051	4054	4073	4075	4092
4094	4099	4100	4102	4104	4125	4127	4157	4159	4177	4178	4193	4195	4286	4288
4333	4337	4354	4358	4371	4372	4383	4384	4392	4395	4416	4418	4430	4431	4436
4437	4444	4448	4488	4489	4532	4534	4547	4550	4574	4577	4624	4625	4632	4633
4647	4652	4658	4664	4666	4677	4680	4683	4684	4686	4688	4695	4699	4704	4706
4710	4801	4802	4803	4809	4819	4827	4832	4833	4835	4843	4846	4853	4854	4855
4877	4944	4945	4948	4961	4962	4970	4972	5012	5033	5034	5036	5045	5094	5095
5100	5110	5142	5192	5195	5198	5240	5277	5286	5356	5441	5442	5506	5509	5645
5647	5790	5792	5844	5846										
625	626	628	643	644	673	674	675	676	677	678	679	680	681	682
701	702	703	704	705	706	707	709	709	710					
971	1014	1124	1459	4943	5034	5280	5861	623	683	711	740	751	762	774
993	594	603	612	614	615	616	617	902	910	912	914	918	925	927
994	797	802	804	808	812	819	892	1004	1007	1010	1011	1012	1013	1016
995	958	959	960	964	965	993	1001	1025	1026	1027	1028	1029	1030	1031
1017	1018	1019	1020	1021	1022	1048	1059	1091	1093	1140	1144	1159	1161	1194
1022	1033	1034	1035	1037	1041	1433	1434	1436	1446	1455	1458	1497	1506	1574
1196	1213	1223	1420	1423	1429	1691	1704	1717	1720	1736	1741	1750	1761	1765
1807	1643	1646	1676	1680	1691	1961	1973	1981	1984	1988	1998	2002	2003	2015
1807	1810	1847	1849	1958	1961	2031	2032	2039	2040	2055	2056	2059	2060	2066
2067	2019	2020	2022	2028	2031	2096	2097	2099	2101	2107	2110	2117	2119	2129
2070	2070	2078	2079	2082	2092	2143	2147	2154	2155	2167	2170	2173	2176	2177
2106	2133	2135	2137	2143	2143	2146	2154	2154	2155	2167	2170	2173	2176	2177
2122	2180	2186	2189	2190	2195	2233	2243	2251	2254	2255	2257	2259	2262	2263
2177	2232	2234	2235	2233	2239	2287	2299	2302	2303	2304	2306	2309	2313	2314
2244	2279	2281	2283	2286	2287	2382	2389	2390	2392	2394	2404	2405	2407	2409
2279	2348	2358	2366	2374	2382	2452	2454	2457	2461	2467	2468	2470	2472	2479
2348	2435	2443	2450	2451	2452	2515	2526	2535	2544	2553	2554	2555	2557	2564
2348	2503	2510	2511	2513	2515	2593	2600	2607	2614	2615	2617	2619	2630	2639
2348	2571	2574	2576	2586	2593	2668	2674	2675	2677	2679	2690	2697	2704	2711
2348	2658	2659	2661	2664	2668	2756	2763	2772	2781	2788	2797	2806	2813	2822
2348	2725	2727	2738	2747	2756	2854	2860	2861	2867	2868	2876	2883	2890	2897
2348	2843	2845	2847	2850	2854	2942	2949	2958	2967	2974	2983	2990	2999	3008
2348	2911	2913	2924	2933	2942	3039	3052	3058	3067	3074	3083	3094	3099	3108
2348	3029	3030	3032	3035	3039	3125	3132	3133	3134	3136	3141	3144	3145	3171
2348	3099	3100	3103	3105	3118	3225	3232	3232	3234	3236	3241	3244	3245	3274
2348	3209	3217	3237	3250	3257	3276	3290	3292	3295	3296	3316	3323	3324	3326
2348	3350	3353	3357	3370	3374	3374	3376	3378	3378	3400	3407	3420	3424	3425
2348	3428	3450	3453	3457	3471	3471	3476	3477	3479	3501	3504	3508	3521	3525
2348	3527	3529	3551	3554	3554	3558	3571	3575	3576	3577	3601	3604	3608	3623
2348	3627	3628	3653	3660	3661	3668	3674	3675	3679	3682	3685	3686	3688	3689
2348	3764	3767	3769	3778	3779	3788	3789	3789	3795	3801	3823	3825	3827	4013
2348	4050	4053	4072	4074	4091	4093	4098	4099	4099	4101	4103	4124	4126	4156
2348	4177	4192	4285	4287	4332	4336	4353	4357	4357	4370	4371	4393	4394	4415
2348	4417	4430	4443	4447	4488	4488	4533	4533	4546	4549	4573	4576	4623	4631
2348	4632	4646	4657	4662	4664	4676	4678	4678	4679	4680	4684	4686	4695	4697
2348	4705	4712	4801	4802	4808	4818	4822	4822	4829	4832	4833	4843	4846	4853
2348	4875	4896	4944	4948	4961	4969	4971	4979	5032	5033	5034	5035	5041	5061

.EQUIV
.EVEN

	5095	5099	5109	5133	5191	5195	5198	5225	5256	5285	5355	5441	5505	5508	5644
.IFF	5646 6122 9655 12224 1736 2003 2056 2108 2156 2199 2254 2366 2450 2510 2572 2658 2722 2842 2908 3017 3085 3144 3296 3407 3521 3627 3802 4102 4371 4574 4855 5286 5459 5599 5774 5958 6102 6270 6496 6657 6812 6912 5023 5500	5789 615 956 1433 1742 2004 2058 2110 2167 2200 2254 2374 2451 2511 2575 2659 2726 2843 2909 3028 3088 3145 3316 3420 3526 3628 3824 4104 4383 4577 4876 5356 4694 4892 5088 1424 1611 1856 3772 4520 4658 4813 4915 5024	5791 616 1038 1498 1751 2015 2059 2111 2170 2202 2255 2382 2452 2514 2577 2660 2728 2845 2912 3029 3090 3171 3323 3425 3527 3629 3826 4125 4392 4624 4896 5441 4833 5032 1455 1612 1860 3775 4521 4693 4825 4928 5034	5843 617 1042 1507 1762 2018 2060 2118 2171 2204 2257 2399 2453 2516 2596 2661 2738 2846 2914 3030 3099 3193 3325 3426 3528 3630 3923 4127 4395 4632 4945 5506 4954 5062 510 1494 1634 1882 3777 4582 4694 4834 4934 5059	5845 623 1049 1575 1766 2019 2067 2119 2174 2205 2258 2390 2454 2526 2592 2662 2747 2847 2924 3031 3100 3209 3326 3427 3529 3639 3928 4157 4416 4633 4954 5509 5066 5070 511 1495 1670 1883 3789 4583 4710 4846 4940 5089	798 1060 1578 1808 2020 2069 2129 2177 2214 2259 2304 2392 2455 2544 2600 2668 2756 2848 2933 3032 3104 3217 3327 3428 3552 3645 4013 4159 4418 4647 4961 5645 5086 5093 512 1517 1814 1928 4184 4618 4777 4871 4984 5105	904 1092 1644 1811 2021 2070 2132 2178 2217 2260 2305 2395 2461 2544 2607 2674 2763 2854 2942 3033 3106 3237 3329 3451 3712 4018 4177 4430 4677 4962 5647 5094 5093 513 1518 1815 1939 4360 4622 4778 4875 4991 5154	809 1094 1647 1848 2022 2079 2133 2179 2219 2262 2306 2404 2467 2553 2614 2676 2772 2860 2949 3039 3108 3250 3351 3457 3571 3729 4051 4193 4436 4680 4972 5790	814 1141 1677 1850 2029 2080 2135 2180 2232 2263 2307 2405 2469 2554 2615 2678 2781 2862 2958 3052 3125 3267 3357 3471 3576 3750 4054 4195 4444 4686 5012 5792	880 1145 1681 1973 2031 2092 2136 2187 2233 2275 2313 2408 2471 2555 2618 2680 2788 2864 2967 3054 3132 3276 3370 3476 3577 3756 4073 4286 4449 4712 5034 5844	910 1160 1691 1974 2032 2096 2137 2189 2234 2278 2343 2410 2473 2556 2620 2690 2797 2866 2974 3056 3133 3290 3375 3477 3578 3759 4075 4288 4488 4801 5036 5846	919 1162 1704 1575 2040 2097 2144 2190 2236 2280 2345 2419 2482 2557 2630 2697 2806 2876 2983 3058 3134 3291 3376 3478 3579 3765 4092 4333 4488 4803 5096	925 1195 1711 1982 2041 2099 2146 2195 2238 2281 2347 2427 2489 2558 2639 2704 2813 2883 2992 3070 3135 3292 3377 3479 3602 3768 4094 4337 4534 4808 5120	928 1197 1718 1988 2052 2100 2147 2197 2239 2282 2349 2435 2496 2564 2648 2711 2822 2890 2999 3077 3136 3293 3378 3502 3608 3789 4099 4354 4822 5110	954 1214 1721 1998 2055 2101 2155 2198 2244 2287 2358 2442 2502 2570 2657 2722 2831 2897 3008 3084 3142 3295 3401 3508 3622 3796 4100 4358 4853 5192
.IPP	774 2452 3527 4362 4612 5265	886 2555 3577 4387 4633 5268	902 2659 3628 4407 4640 5275	1037 2845 3759 4412 4662 5299	1973 3030 4058 4422 4712 5340	2020 3134 4069 4440 4818 5451	2058 3219 4076 4449 4853 5493	2099 3240 4087 4468 5052	2135 3258 4160 4476 5082	2177 3271 4188 4503 5199	2198 3290 4199 4513 5200	2232 3326 4215 4522 5221	2257 3376 4280 4535 5252	2281 3426 4311 4542 5253	2304 3477 4318 4586 5257
.LIST	1419 2135	1425 2147	1436 2177	1455 2190	1459 2198	1957 2205	1963 2232	1973 2239	1988 2257	2020 2263	2032 2281	2058 2287	2070 2304	2099 2313	2111 2452

	2461	2555	2564	2659	2668	2845	2854	3030	3039	3134	3145	3290	3296	3326	3357
	3376	3407	3426	3457	3477	3508	3527	3558	3577	3608	3628	3645	3760	4657	4846
	4961														
.MACRO	1	592	617	918	1973	2198	3134	3326	3376	3426	3477	3527	3577	3628	4646
.MCALL	592	725	965	1436											
.MEXIT	1036														
.MLIST	1	592	604	616	725	774	781	918	954	955	356	957	958	965	968
	1419	1425	1436	1455	1459	1957	1963	1973	1988	2020	2032	2058	2070	2099	2111
	2135	2147	2177	2190	2198	2205	2232	2239	2257	2263	2281	2287	2304	2313	2323
	2461	2555	2564	2659	2668	2845	2854	3030	3039	3134	3145	3290	3296	3326	3357
	3376	3407	3426	3457	3477	3508	3527	3558	3577	3608	3628	3645	3760	4657	4946
	4961														
.PAGE	807	879	918	964	1740	1846	2020	2257	2304	2452	2555	2659	3030	3134	3290
	3376	3426	3477	3527	3577	3628	3726	3749	4049	4101	4155	4352			
.REM	1														
.REPT	781	954	1063	1420	1423	1958	1961								
.SBTTL	605	621	728	775	785	799	881	920	967	1039	1061	1193	1212	1277	1418
	1642	1675	1740	1846	1972	1973	2020	2058	2099	2135	2172	2177	2198	2232	2257
	2281	2304	2452	2555	2659	2845	3030	3134	3290	3325	3326	3375	3426	3477	3527
	3577	3628	3726	3751	3793	3794	4049	4155	4352	4545	4648	4804	4856	4946	5037
	5111	5193	5287	5357	5446	5504	5643								
.TITLE	593														
.WORD	781	782	783	794	795	805	876	877	878	911	913	927	930	931	932
	933	936	937	938	939	940	941	942	943	944	945	954	955	956	957
	973	974	975	976	977	978	979	980	984	985	986	999	1003	1006	1009
	1015	1016	1017	1018	1019	1020	1021	1022	1023	1024	1025	1026	1027	1028	1029
	1030	1031	1032	1033	1034	1053	1054	1055	1056	1057	1058	1063	1064	1065	1066
	1067	1068	1069	1070	1071	1072	1073	1074	1075	1076	1077	1078	1079	1080	1081
	1082	1083	1084	1085	1086	1087	1094	1095	1097	1098	1100	1101	1103	1104	1107
	1108	1110	1111	1113	1114	1116	1117	1118	1125	1127	1128	1130	1131	1133	1134
	1136	1137	1138	1191	1456	1494	1517	1562	1570	1588	1601	1611	1633	1670	1691
	1704	1711	1736	1814	1820	1828	1855	1882	1927	1939	1984	1998	2015	2031	2052
	2069	2092	2110	2129	2146	2167	2189	2204	2214	2238	2251	2262	2275	2286	2299
	2309	2358	2366	2374	2382	2419	2427	2435	2443	2457	2482	2489	2496	2503	2526
	2535	2544	2560	2586	2593	2600	2607	2630	2639	2648	2664	2690	2697	2704	2711
	2738	2747	2756	2763	2772	2781	2788	2797	2806	2813	2822	2831	2950	2876	2983
	2890	2897	2924	2933	2942	2949	2958	2967	2974	2983	2992	2999	3008	3017	3035
	3070	3077	3118	3125	3144	3171	3193	3209	3217	3237	3250	3267	3276	3295	3316
	3353	3370	3403	3420	3453	3471	3504	3521	3554	3571	3604	3622	3641	3712	3729
	3748	3764	3767	3770	3777	4177	4184	4230	4231	4259	4260	4296	4297	4306	4307
	4360	4371	4378	4383	4430	4436	4457	4488	4495	4520	4582	4603	4610	4733	4777
	4825	4834	4864	4887	4889	4893	4895	4934	4940	4984	4991	4997	5002	5008	5010
	5014	5018	5023	5089	5091	5093	5106	5139	5185	5223	5346	5417	5433	5436	5441
	5489	5499													

ERRORS DETECTED: 0
DEFAULT GLOBALS GENERATED: 0

*DZQMCB, DZQMCB, SEQ/SOL/CRF/DS:ERFZ/EN:ABS=DSKM:SYSMAC.SML, DSKM:DZQMCB.P11
RUN-TIME: 74 82 12 SECONDS
RUN-TIME RATIO: 311/171=1.8
CORE USED: 36K (71 PAGES)

