

801

.REM %

IDENTIFICATION

PRODUCT CODE: MAINDEC-11-DCMSB-B-D  
PRODUCT NAME: MOS SEMICONDUCTOR MEMORY DIAGNOSTIC  
DATE CREATED: 23-APR-1973  
MAINTAINER: DIAGNOSTIC GROUP  
AUTHOR: SUB MALLICK

COPYRIGHT (C) 1973  
DIGITAL EQUIPMENT CORPORATION  
MAYNARD, MASSACHUSETTS 01754

MEM MOS MEMORY MACY: 27.732) 22-SEP-76 14:48 PAGE 1  
DCMSB.P1:

29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84

1.0 ABSTRACT

THIS PROGRAM IS TO BE USED TO TEST MOS SEMICONDUCTOR MEMORY ONLY AFTER ALL OTHER MEMORY DIAGNOSTICS HAVE BEEN RUN. THIS PROGRAM ONLY CHECKS REFRESH CYCLE PROBLEMS OF MOS MEMORY ON PDP-11/45.

2.0 REQUIREMENTS

2.1 EQUIPMENT

PDP-11/45 STANDARD COMPUTER WITH MOS SEMICONDUCTOR MEMORY.

2.2 STORAGE

THE ROUTINE USES MEMORY 0 TO 11510

3.0 LOADING PROCEDURE

PROCEDURE FOR NORMAL ABSOLUTE TAPES SHOULD BE FOLLOWED.

3.0.1 RELOCATION PROCEDURE

THIS PROGRAM IS RELOCATABLE. FOLLOW NORMAL RELOCATION PROCEDURE. THAT IS SET SWITCH "0". TO INDICATE THAT RELOCATION IS REQUIRED. THEN SET THE SWITCHES WHERE THE PROGRAM IS TO START LOADING. THEN HIT "START" TO LOAD TAPE. ALL PROGRAM ADDRESSES BECOME INCREASED BY THIS RELOCATION FACTOR

FOR EXAMPLE IF SWITCH 12 AND 0 WAS SET BEFORE LOADING THEN 40000 IS THE RELOCATION FACTOR AND HENCE 40204 IS THE STARTING ADDRESS

4.0 STARTING PROCEDURE

4.1 CONTROL SWITCH SETTINGS

BASIC SWITCH REGISTER SETTINGS ARE:

SWITCH	USE
15	HALT ON ERROR
14	0 - DISABLE PARITY CHECKING 1 - ENABLE PARITY CHECKING
13	INHIBIT ERROR TYPEOUTS

95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140

- 12 0 - SET PROGRAM TO CATCH BOTH BIT DROPS AND PICKUPS
- 11 1 - SET PROGRAM AS INDICATED BY SWITCH 11
- 10 0 - SET PROGRAM TO CATCH BIT DROPS
- 9 1 - SET PROGRAM TO CATCH BIT PICKUPS
- 8 0 - PRINT END PASS
- 7 1 - BELL ON ERROR
- 6 0 - PRINT FIRST ERROR PER PRECHARGE
- 5 1 - PRINT ALL ERRORS PER PRECHARGE
- 4 0 - INHIBIT PRINTING DATA HEADER
- 3 0=TEST READ AND WRITE FROM OUTSIDE OF THE CHIP
- 2 1=TEST INSTRUCTIONS IN THE CHIP
- 1 1 LOCK ON PRESENT MOD
- 0 0 LOCK ON PRESENT PRECHARGE

4.2 STARTING ADDRESSES

- 200 = STARTING ADDRESS IF ALL OF MEMORY IS TO BE TESTED AND NO LIMITS CHOSEN
- 204 = STARTING ADDRESS IF LIMITS ARE TO BE CHOSEN
- 210 = RESTART ADDRESS WITH LAST SPECIFIED LIMITS

4.3 PROGRAM AND/OR OPERATOR ACTION

4.3.1 OPERATOR ACTION

THERE ARE TWO MODES OF OPERATION FOR THIS PROGRAM  
 A) NO LIMITS CHOSEN  
 B) LIMITS CHOSEN

A) NO LIMITS CHOSEN

FOR THIS MODE OF OPERATION RELOCATION CANNOT BE USED THAT IS BEFORE LOADING TAPE, SWITCH 0 MUST BE 0. STARTING ADDRESS IS 200. THE MEMORY SIZE IS PRINTED IN OCTAL THE PROGRAM TYPES OUT "SET SWITCHES FOR OPERATING MODE. REFER SECTION 4.1

IE. PICKUPS OR DROPS OR BOTH THEN HIT CONTINUE " ON SUBSEQUENT STARTS THE TYPEOUT IS SHORTER. NOW CHOOSE ALL SWITCH SETTINGS. SWITCH 11 AND 12 MUST BE LEFT ON AS DESIRED AND IF CHANGED DURING OPERATION NO CHANGE WILL TAKE PLACE TILL THE "END PASS" PRINT OUT. THEN ALL OF MEMORY IS CHECKED STARTING FROM THE SECOND 4K TILL THE LAST ADDRESS IS REACHED. THEN THE PROGRAM IS MOVED INTO THE SECC:0 4K AND THE FIRST 4K IS CHECKED. THIS MODE OF OPERATION DESTROYES THE LOADER AND IF THE PROGRAM HAS MOVED INTO THE SECOND 4K AND AN OPERATOR HALT IS PERFORMED THEN RESTART ADDRESS IS NOT 200 BUT

141  
142  
143  
144  
145  
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

40200.

## B) LIMITS CHOSEN

IN THIS MODE OF OPERATION RELOCATION CAN BE USED. TO RELOCATE PROGRAM REFER TO SECTION 3.0.1. STARTING ADDRESS IS 204 OR RELOCATION FACTOR PLUS 204. THE MEMORY SIZE IS PRINTED IN OCTAL THEN THE PROGRAM TYPES "TYPE "STARTING BANK" NUMBER AND CARRIAGE RETURN. REFER TO SECTION 4.3.1 BEFORE CHOOSING" OR SHORTER TYPE OUTS ON SUBSEQUENT STARTS THIS IS THE SECTION REFERED TO

TYPE THE 4K BANK WHERE YOU WISH TO BEGIN TESTING

TYPE	TO START AT	TYPE	TO START AT
0	000000 (0K)	20	400000 (64K)
1	020000 (4K)	21	420000 (68K)
2	040000 (8K)	22	440000 (72K)
3	060000 (12K)	23	460000 (76K)
4	100000 (16K)	24	500000 (80K)
5	120000 (20K)	25	520000 (84K)
6	140000 (24K)	26	540000 (88K)
7	160000 (28K)	27	560000 (92K)
10	200000 (32K)	30	600000 (96K)
11	220000 (36K)	31	620000 (100K)
12	240000 (40K)	32	640000 (104K)
13	260000 (44K)	33	660000 (108K)
14	300000 (48K)	34	700000 (112K)
15	320000 (52K)	35	720000 (116K)
16	340000 (56K)	36	740000 (120K)
17	360000 (60K)		

TYPE ONLY NUMBERS SHOWN !!!

MEMORY CAN ONLY BE TESTED IN 4K GROUPS STARTING AND FINISHING AT 4K BOUNDARY  
FOR EXAMPLE IF YOU WISH TO TEST THE THIRD 4K BANK (STARTING ADDRESS 40000) TYPE "2".

THEN THE PROGRAM TYPES "TYPE NUMBER OF 4K BANKS TO BE TESTED AND CARRIAGE RETURN". ON SUBSEQUENT STARTS TYPEOUT IS SHORTER. AFTER TYPING THE APPROPRIATE NUMBER AND CARRIAGE RETURN THE PROGRAM TYPES " SET SWITCHES FOR OPERATING MODE. REFER SECTION 4.1 I.E. PICKUPS OR DROPS OR BOTH THEN HIT CONTINUE" ON SUBSEQUENT STARTS TYPEOUTS ARE SHORTER. THIS MEANS NOW IS THE TIME TO CHOOSE ALL SWITCH SETTINGS. SWITCH 11 AND 12 MUST BE LEFT ON AS DESIRED AND IF CHANGED DURING OPERATION NO CHANGE WILL TAKE PLACE TILL THE "END PASS" PRINT OUT. CHECKING IS NOW STARTED

IF STARTING BANK AND NUMBER OF BANKS ARE INCORRECTLY CHOSEN THEN RUBOUT MAY BE HIT TO DELETE THE LAST CHARACTER IF CARRIAGE RETURN HAS NOT ALREADY BEEN HIT. IF CARRIAGE RETURN

197  
198  
199  
200  
201  
202  
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

HAS BEEN HIT THEN RESTARTING IS NECESSARY. IF NO CORRECTION IS MADE THEN THE PROGRAM MAY DESTROY ITSELF OR IT MAY TIME OUT ON A NON EXISTANT MEMORY.

IN ORDER TO TEST THE FIRST 4K OF MEMORY WITH 204 STARTING THAT IS BY CHOOSING LIMITS RELOCATE THE PROGRAM BY LOADING IT INTO A HIGHER AREA OF MEMORY SAY AT 40000 AND TEST THE FIRST FOUR K BY MAKING THE STARTING BANK "0" AND NUMBER OF BANKS "1"

4.3.2 PROGRAM ACTION

THE PROGRAM WILL FIRST FILL A 4K BLOCK WITH WHAT EVER IS IN "BCKGRD" .THEN THE PROGRAM WILL WRITE "TSTPAT" INTO ONE LOCATION CALLED TSTPAT . IN ANOTHER LOCATION CALLED TEST-LOCATION IT WILL WRITE A CLR RD (OR ALL "0" OR ALL "1" DEPENDING ON SWITCHES 7 AND 12. LOC TSTPAT AND TEST-LOCATION WILL HAVE BIT 8 OF OPPOSITE VALUE. THE TEST-LOCATION WILL BE OPERATED ON 1000 TIMES THEN THE 4K WILL BE CHECKED. THEN ONE WILL BE ADDED TO LOCATION TSTPAT AND THE PROCESS REPEATED TILL ONE 4K IS DONE THEN THE NEXT 4K WILL BE STARTED.

5.0 OPERATING PROCEDURE

5.1 OPERATIONAL SWITCH SETTINGS

SEE 4.1

6.0 ERRORS

6.1 ERROR PRINTOUT

THERE ARE THREE TYPES OF ERROR PRINTOUTS

- A) BACK GROUND CHANGED
- B) INSTRUCTION CHANGED
- C) TEST PATTERN

EXAMPLE

BACK GROUND CHANGED						
ERROR	BCKGRD	EXPECT	RCV'D			
PC	ADDRESS	DATA	DATA	COL	ROW	PRECHARGE
103166	006500	177777	177577	6	23	0

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

THE COL=COLIMN AND ROW PRINTOUTS ARE FOR ENGINEERING USE WHICH TELLS WHAT BIT WITHIN A CHIP IS FAULTY. IF INFORMATION WITHIN A CHIP IS NOT NECESSARY DISREGARD THESE TWO (COL, ROW)

6.2 ERROR RECOVERY

IN GENERAL, TEST FAILURES WILL PRINTOUT AN ERROR MESSAGE AND CONTINUE. IF THE "HALT ON ERROR" SWITCH IS SET HITTING CONTINUE WILL RECOVER. IF THE PROGRAM HANGS UP IN A LOOP, THE ERROR IS LIKELY TO BE A SIGNAL WHICH WAS NEVER RECEIVED. IF A HALT OCCURS IN THE TRAP AND VECTOR AREA THE PROGRAM MUST BE RESTARTED. IF THE PROGRAM HALTS IN THE MAIN FLOW, CONSULT THE LISTING IF NO MESSAGE IS TYPED OUT.

7.0 RESTRICTIONS

RELOCATION CANNOT BE DONE INTO 160000  
IF PARITY CHECKING IS ENABLED THEN ON AN ERROR IT IS NOT GUARANTEED WHICH WILL SHOW UP: THE PARITY ERROR OR THE BIT ERROR.  
IT IS RECOMMENDED THAT PARITY IS ENABLED ONLY WHEN LOOKING FOR PARITY ERRORS.

8.0 MSCCELLANEOUS

8.1 EXECUTION TIME

APPROXIMATELY 10 MIN PER 4K TESTED. (BOTH PICKUPS & DROPS.)

```
%
.ENABL ABS
.TITLE TEST MOS MEMORY
;COPYRIGHT 1973 DIGITAL EQUIPMENT CORP., MAYNARD, MASS. 01754
;PROGRAM BY JIM LACEY, SUB MALLICK
;*****
```

;OPERATIONAL SWITCH SETTINGS

SWITCH	USE
15	HALT ON ERROR
14	0 - DISABLE PARITY CHECKING 1 - ENABLE PARITY CHECKING
13	INHIBIT ERROR TYPEOUTS
12	0 - SET PROGRAM TO CATCH BOTH BIT DROPS AND PICKUPS 1 - SET PROGRAM AS INDICATED BY SWITCH 11
11	0 - SET PROGRAM TO CATCH BIT DROPS 1 - SET PROGRAM TO CATCH BIT PICKUPS
10	0 - PRINT END PASS

```

309          :
310          :           9          1 - BELL ON ERROR
311          :           :          0 - PRINT FIRST ERROR PER PRECHARGE
312          :           :          1 - PRINT ALL ERRORS PER PRECHARGE
313          :           8          INHIBIT PRINTING DATA HEADER
314          :           7          0=TEST READ AND WRITE FROM OUTSIDE OF THE CHIP
315          :           :          1=TEST INSTRUCTIONS IN THE CHIP
316          :           1          LOCK ON PRESENT MOD
317          :           0          LOCK ON PRESENT PRECHARGE
318          : *****

```

```

320          ;BASIC DEFINITIONS
321          .EQUIV EMT,HLT          ;BASIC DEFINITION OF ERROR CALL
322          .EQUIV IOT,SCOPE       ;BASIC DEFINITION OF SCOPE CALL
323          PS= 177776             ;PROCESSOR STATUS WORD
324          .EQUIV PS,PSW
325          SWR= 177570           ;SWITCH REGISTER
326          DISPLAY=SWR

```

```

327          ;REGISTER DEFINITION
328          R0= %0                 ;GENERAL REGISTER
329          R1= %1                 ;GENERAL REGISTER
330          R2= %2                 ;GENERAL REGISTER
331          R3= %3                 ;GENERAL REGISTER
332          R4= %4                 ;GENERAL REGISTER
333          R5= %5                 ;GENERAL REGISTER
334          R6= %6                 ;GENERAL REGISTER
335          R7= %7                 ;GENERAL REGISTER
336          .EQUIV R6,SP          ;STACK POINTER
337          .EQUIV R7,PC          ;PROGRAM COUNTER

```

```

340          ;SWITCH DEFINITION
341          SW15= 100000
342          SW14= 40000
343          SW13= 20000
344          SW12= 10000
345          SW11= 4000
346          SW10= 2000
347          SW09= 1000
348          SW08= 400
349          SW07= 200
350          SW06= 100
351          SW05= 40
352          SW04= 20
353          SW03= 10
354          SW02= 4
355          SW01= 2
356          SW00= 1
357          .EQUIV SW09,SW9
358          .EQUIV SW08,SW8
359          .EQUIV SW07,SW7
360          .EQUIV SW06,SW6
361          .EQUIV SW05,SW5
362          .EQUIV SW04,SW4
363          .EQUIV SW03,SW3
364          .EQUIV SW02,SW2

```

TEST MOS MEMORY MACY11 27(732) 22-SEP-76 14:48 PAGE 8  
DCMS88.P11

365		.EQUIV SW01,SW1
366		.EQUIV SW00,SW0
367		
368		: MISCELLANEOUS BIT ASSIGNMENT
369	100000	BIT15= 100000
370	040000	BIT14= 40000
371	020000	BIT13= 20000
372	010000	BIT12= 10000
373	004000	BIT11= 4000
374	002000	BIT10= 2000
375	001000	BIT09= 1000
376	000400	BIT08= 400
377	000200	BIT07= 200
378	000100	BIT06= 100
379	000040	BIT05= 40
380	000020	BIT04= 20
381	000010	BIT03= 10
382	000004	BIT02= 4
383	000002	BIT01= 2
384	000001	BIT00= 1
385		.EQUIV BIT09,BIT9
386		.EQUIV BIT08,BIT8
387		.EQUIV BIT07,BIT7
388		.EQUIV BIT06,BIT6
389		.EQUIV BIT05,BIT5
390		.EQUIV BIT04,BIT4
391		.EQUIV BIT03,BIT3
392		.EQUIV BIT02,BIT2
393		.EQUIV BIT01,BIT1
394		.EQUIV BIT00,BIT0
395		
396		: VECTOR ADDRESSES
397	000004	ERRVEC= 4
398	000010	RESVEC= 10
399	000014	TBITVEC=14
400	000014	TRTVEC= 14
401	000014	BPTVEC= 14
402	000020	IOTVEC= 20
403	000024	PWRVEC= 24
404	000030	EMTVEC= 30
405	000034	TRAPVEC=34
406	000114	PARVEC=114
407	UJ0250	MMVEC=250



408  
409  
410 000100 000100 FIRST: .=100  
411 000000 .WORD 0

;KT11-D STATUS REGISTER ADDRESSES

412  
413  
414  
415  
416 000102 177572 SR0: 177572  
417 000104 177574 SR1: 177574  
418 000106 177576 SR2: 177576  
419 000110 172516 SR3: 172516  
420  
421

;KERNAL PAGE DESCRIPTOR REGISTERS

422  
423  
424  
425 000112 172300 KIPDR0: 172300  
426 000114 172302 KIPDR1: 172302  
427 000116 172304 KIPDR2: 172304  
428 000120 172306 KIPDR3: 172306  
429 000122 172310 KIPDR4: 172310  
430 000124 172312 KIPDR5: 172312  
431 000126 172314 KIPDR6: 172314  
432 000130 172316 KIPDR7: 172316  
433

;KERNAL PAGE ADDRESS REGISTERS

434  
435  
436 000132 172340 KIPAR0: 172340  
437 000134 172342 KIPAR1: 172342  
438 000136 172344 KIPAR2: 172344  
439 000140 172346 KIPAR3: 172346  
440 000142 172350 KIPAR4: 172350  
441 000144 172352 KIPAR5: 172352  
442 000146 172354 KIPAR6: 172354  
443 000150 172356 KIPAR7: 172356  
444

;KT11 VECTOR ADDRESS

445  
446  
447 000152 000250 000252 SEGVEC: 250,252

448  
 449 000200 000167 003250  
 450  
 451 000204 000167 002536  
 452 000210 000167 003666  
 453  
 454  
 455  
 456  
 457 000214 000000  
 458 000216 000000  
 459  
 460  
 461

=200  
 JMP BEGN11  
 JMP BEGN1  
 JMP BEGN3

;JUMP TO START TO TEST ALL OF  
 ;MEMORY WITHOUT CHOOSING LIMITS  
 ;JUMP TO START CHOOSE LIMITS  
 ;JUMP TO RESTART USING LAST SPECIFIED LIMITS

IN2AC: 0  
 FRSTAD: 0

462 000300 . =300

463  
464  
465 000300 016701 002364  
466 000304 016702 005022  
467 000310 016703 005014  
468 000314 012122  
469 000316 077302  
470 000320 000137 000204

RESTOR: MOV T01,R1  
MOV FROM,R2  
MOV LENGTH,R3  
1\$: MOV (R1)+,(R2)+  
SOB R3,1\$  
JMP @#204

471  
472  
473 ;\*\*\*\*\*  
474

475  
476  
477  
478  
479 000324 010704  
480 000326 062704 011164  
481 000332 016702 002334  
482 000336 012701 003132  
483 000342 012442  
484 000344 077102  
485 000346 004767 006666  
486 000352 011466  
487 000354 000000

LOADER: MOV PC,R4  
ADD #LAST+2-.,R4  
MOV LODER,R2  
MOV #17774-LAST/2,R1  
1\$: MOV (R4)+,-(R2)  
SOB R1,1\$  
JSR PC,TYPE  
FIN  
HALT

488  
489  
490



```

547      :CALL:
548      :      JSR      PC,$SIZE
549      :      RETURN
550      :$LSTAD WILL CONTAIN THE LAST VIRTUAL ADDRESS OF THE LAST BANK
551      :$LSTBK WILL CONTAIN THE LAST BANK AS A SAF
552      :$KT11 IS THE MEMORY MANAGEMENT KEY
553      :BIT07 = 0 DON'T USE MEMORY MANAGEMENT
554      :BIT15 = 0 DON'T HAVE MEMORY MANAGEMENT OPTION
555
556 001300 010546      $SIZE:  MOV      R5,-(SP)          ;SAVE R5 ON THE STACK
557 001302 010446      MOV      R4,-(SP)
558 001304 010346      MOV      R3,-(SP)
559 001306 010146      MOV      R1,-(SP)          ;SAVE R1 ON THE STACK
560 001310 010046      MOV      R0,-(SP)          ;SAVE R0 ON THE STACK
561 001312 010605      MOV      SP,R5            ;SAVE THE STACK POINTER
562 001314 012701 003776  MOV      #3776,R1
563 001320 0105727      MOV      (PC)+
564 001322 000000      $KT11:  0                ;USE MEMORY MANAGEMENT?
565 001324 100152      BPL      $SCORE          ;BR IF NO
566 001326 010746      $SEG:  MOV      PC,-(SP)
567 001330 062716 000322  ADD      #$SCORE-,(SP)
568 001334 012637 000004  MOV      (SP)+,$ERRVEC    ;SET FOR TIMEOUT
569 001340 005777 176536  TST      $SR0            ;KT11 ARE YOU THERE?
570 001344 052767 100000 177750  BIS      #BIT15,$KT11    ;YES--SET KT11 KEY
571 001352 012703 077406  MOV      #77406,R3
572 001356 010377 176530  MOV      R3,$KIPDR0
573 001362 010377 176526  MOV      R3,$KIPDR1
574 001366 010377 176524  MOV      R3,$KIPDR2
575 001372 010377 176522  MOV      R3,$KIPDR3
576 001376 010377 176520  MOV      R3,$KIPDR4
577 001402 010377 176516  MOV      R3,$KIPDR5
578 001406 010377 176514  MOV      R3,$KIPDR6
579 001412 010377 176512  MOV      PC,$KIPDR7
580
581
582 001416 005077 176510  CLR      $KIPAR0
583 001422 012777 000200 176504  MOV      #200,$KIPAR1
584 001430 012777 000400 176500  MOV      #400,$KIPAR2
585 001436 012777 000600 176474  MOV      #600,$KIPAR3
586 001444 012777 001000 176470  MOV      #1000,$KIPAR4
587 001452 012777 001200 176464  MOV      #1200,$KIPAR5
588 001460 012777 001400 176460  MOV      #1400,$KIPAR6
589 001466 012777 007600 176454  MOV      #7600,$KIPAR7
590
591
592 001474 010703      MOV      PC,R3
593 001476 000241      CLC
594 001500 006103      ROL      R3
595 001502 006103      ROL      R3
596 001504 006103      ROL      R3
597 001506 006103      ROL      R3
598 001510 042703 177770  BIC      #177770,R3
599 001514 022703 000006  CMP      #6,R3
600 001520 001403      BEQ      2$
601
602

```

```

603 001522 016704 176420      MOV      KIPAR6,R4
604 001526 000425      SR      35
605
606
607
608 001530 016704 176410      25:     MOV      KIPAR5,R4
609 001534 012767 123776 000070      MOV      #123776,15+2
610 001542 012703 120000      MOV      #120000,R3
611 001546 010367 003354      MOV      R3,SEGO+2
612 001552 010367 003356      MOV      R3,SEGO+10
613 001556 010367 003376      MOV      R3,SEG10+2
614 001562 010367 003400      MOV      R3,SEG10+10
615 001566 012767 137776 003346      MOV      #137776,SEGO+16
616 001574 012767 137776 003372      MOV      #137776,SEG10+16
617
618
619
620 001602 010467 001060      35:     MOV      R4,PAGE
621 001606 005014      CLR      (R4)
622
623
624
625
626 001610 012777 000001 176264      MOV      #1,JSRO                ;TURN ON MEMORY MANAGEMENT
627 001616 010746      MOV      PC,-(SP)
628 001620 062716 000026      ADD      #SKTOUT-(SP)
629 001624 012637 000004      MOV      (SP)+,2#ERRVEC ;SET FOR TIME OUT
630 001630 005737 143776      15:     TST      2#143776                ;TRAP ON NON-EX-MEM
631 001634 062714 000040      ADD      #40,(R4)                ;MAKE A 1K STEP
632 001640 027714 176304      CMP      #KIPAR7,(R4)            ;LAST ONE?
633 001644 003371      BGT      15                      ;NO--TRY IT
634 001646 011400      SKTOUT: MOV      (R4),R0            ;GET LAST BANK+1
635 001650 000420      BR
636 001652 010746      SCORE:  MOV      PC,-(SP)
637 001654 062716 000032      ADD      #SCOROUT-(SP)
638 001660 012637 000004      MOV      (SP)+,2#ERRVEC ;SET FOR TIMEOUT
639 001664 005000      CLR      R0                      ;SET UP BANK
640 001666 062701 004000      15:     ADD      #4000,R1                ;INCREMENT BY 1K
641 001672 062700 000040      ADD      #40,R0                  ;1K STEP
642 001676 005711      TST      (R1)                    ;TRAP ON TIME OUT
643 001700 022701 177776      CMP      #177776,R1              ;LAST ONE
644 001704 001370      BNE      15                      ;NO--TRY AGAIN
645 001706 162701 004000      SCOROUT: SUB      #4000,R1
646 001712 162700 000040      SSIZEX: SUB      #40,R0
647 001716 012737 000006 000004      MOV      #6,2#ERRVEC
648 001724 010506      MOV      R5,SP
649 001726 010167 000032      MOV      R1,SLSTAD
650 001732 010067 000030      MOV      R0,SLSTBK
651 001736 005767 177360      TST      SKT11
652 001742 100002      BPL      15
653 001744 005077 176132      CLR      JSRO
654 001750 012600      15:     MOV      (SP)+,R0
655 001752 012601      MOV      (SP)+,R1
656 001754 012603      MOV      (SP)+,R3
657 001756 012604      MOV      (SP)+,R4
658 001760 012605      MOV      (SP)+,R5                ;RESTORE R5

```

659 001762 000207  
660 001764 000000  
661 001766 000000

RTS PC  
\$LSTAD: .WORD 0  
\$LSTBK: .WORD 0

:CONTAINS THE LAST ADDRESS  
:CONTAINS THE LAST BANK

```

662 ;ROUTINE TO GET TYPED ASCII DATA AND CONVERT TO OCTAL.
663 ;LEAVES THE OCTAL DATA IN THE ADDRESS FOLLOWING THE CALL.
664 ;CALL: RECD ;(A TRAP TYPE INST.)
665 ;.WORD 0 ;CONTAINS RETURNED DATA
666 ; ;RETURNS HERE
667 ;CONTINUE
668
669 001770 177560 TKS: 177560
670 001772 177562 TKB: 177562
671
672
673 001774 000240 .RECD: NOP
674 001776 005046 CLR -(SP)
675 002000 105777 177764 1$: TSTB @TKS ;WAIT FOR USER RESPONSE
676 002004 100375 BPL 1$
677 002006 117746 177760 MOVB @TKB, -(SP) ;GET ASCII CHARACTER
678 002012 042716 177600 BIC #177600, (SP) ;
679 002016 122716 000177 CMPB #177, (SP) ;CHECK IF RUBOUT
680 002022 001013 BNE 2$ ;BRANCH IF NOT RUBOUT
681 002024 012767 000134 000142 MOV #' \, 4$
682 002032 004767 005202 JSR PC, TYPE ;ECHO BACK SLASH
683 002036 002174 4$
684 002040 005726 6$: TST (SP)+ ;POP LAST TYPED CHAR OFF THE STACK
685 002042 006016 ROR (SP) ;SHIFT LAST TYPD CHARACTER
686 002044 006216 ASR (SP)
687 002046 006216 ASR (SP)
688 002050 000753 BR 1$ ;AND WAIT FOR NEXT CHAR.
689
690
691 002052 122716 000015 2$: CMPB #15, (SP) ;CHECK IF CARRIAGE RETURN
692 002056 001011 BNE 3$ ;BRANCH IF NOT A CARRIAGE RETURN
693 002060 004767 005154 JSR PC, TYPE
694 002064 002647 $CRLF
695 002066 005726 TST (SP)+ ;POP CARRIAGE RETURN OFF THE STACK
696 002070 012676 000000 MOV (SP)+, @ (SP) ;PUT DATA IN ADDRESS FOLLOWING CALL
697 002074 062716 000002 ADD #2, (SP)
698 002100 000207 RTS PC
699 002102 111667 000066 3$: MOVB (SP), 4$
700
701 002106 004767 005126 JSR PC, TYPE ;GO TO PRINT ROUTINE
702 002112 002174 4$ ;ECHO CHARACTER
703 002114 042767 177707 000052 BIC #177707, 4$
704 002122 022767 000060 000044 CMP #60, 4$
705 002130 001407 BEQ 5$
706 002132 004767 005102 JSR PC, TYPE
707 002136 002647 $CRLF
708 002140 004767 005074 JSR PC, TYPE
709 002144 011217 SORRY
710 002146 000734 BR 6$
711 002150 042716 177770 5$: BIC #177770, (SP) ;SAVE ONLY 3 MSBS
712 002154 006366 000002 ASL 2(SP) ;SHIFT LAST CHARACTER OVER
713 002160 006366 000002 ASL 2(SP) ;THREE PLACES
714 002164 006366 000002 ASL 2(SP)
715 002170 052616 BIS (SP)+, (SP) ;INSET OCTAL CHAR
716 002172 000702 BR 1$ ;GO WAIT FOR NEXT CHAR.
717 002174 000000 4$: .WORD 0 ;CONTAINS CHARACTER TO BE TYPED

```



```

718                                     :ROUTINE TO SET PARITY ENABLE ON MA/MF PARITY MEMORIES
719                                     PARCSR= 172100 ;ADDRESS OF FIRST POSSIBLE PARITY REG.
720
721
722
723 002176 012737 002254 000114 MAMF:  MOV    #PARSR, 2#PARVEC      ;LOAD VECTOR
724 002204 012737 000340 000116      MOV    #340, 2#PARVEC+2    ;AND PRIORITY LEVEL
725 002212 012737 000006 000004      MOV    #ERRVEC+2, 2#ERRVEC ;SET TIME OUT TRAP VECTOR
726 002220 012737 000002 000006      MOV    #RTI, 2#ERRVEC+2   ;DO RTI ON TIMEOUT TRAP
727 002226 012700 172100      MOV    #PARCSR, R0       ;GET FIRST POSSIBLE ADDRESS
728 002232 012702 000001      MOV    #1, R2           ;SET REG. COUNTER
729
730 002236 012720 000001      IS:   MOV    #1, (R0)+    ;SET ACTION ENABLE IF THERE
731 002242 005302      ASL    R2              ;CHECK IF 16 REG. HAVE BEEN DONE
732 002244 103374      BCC    1$             ;REPEAT IF 16 NOT ENABLED
733 002246 005037 000006      CLR    2#ERRVEC+2     ;RESTORE HALT ON TIMEOUT
734 002252 000207      RTS    PC             ;RETURN
735
736

```





803  
804  
805  
806  
807  
808 002540 000000  
809 002542 000000  
810 002544 000000  
811 002546 000000  
812 002550 000000  
813 002552 000000  
814 002554 000  
815 002555 000  
816 002556 000000 000000  
817 002562 000  
818 002563 000  
819 002564 000000  
820 002566 000000  
821 002570 000000  
822 002572 000000  
823 002574 000000  
824 002576 000000  
825 002600 000070  
826 002602 000000  
827 002604 000000  
828 002606 000000  
829 002610 017776  
830 002612 000000  
831 002614 000000  
832 002616 000000  
833 002620 000000  
834 002622 027420  
835 002624 177777  
836 002626 177777  
837 002630 000000  
838 002632 000400  
839 002634 000000  
840 002636 000000  
841 002640 000000  
842 002642 000000  
843 002644 000207  
844 002646 077  
845 002647 015  
846 002650 000012  
847 002652 000000  
848 002654 000000  
849 002656 177777  
850 002660 000000  
851 002662 000000  
852 002664 000000  
853 002666 000000  
854 002670 000000  
855 002672 000000

\*\*\*\*\*

:COMMON TAGS

\$PASS: .WORD 0 ;CONTAINS PASS COUNT  
\$STNM: .WORD 0 ;CONTAINS THE TEST NUMBER  
\$ICNT: .WORD 0 ;CONTAINS SUBTEST ITERATION COUNT  
\$LPADR: .WORD 0 ;CONTAINS SCOPE LOOP ADDRESS  
\$LPERR: .WORD 0 ;CONTAINS SCOPE RETURN FOR ERRORS  
\$ERTTL: .WORD 0 ;CONTAINS TOTAL ERRORS DETECTED  
\$ERFLG: .BYTE 0 ;CONTAINS ERROR FLAG  
;RESERVED--NOT TO BE USED  
;RESERVED--NOT TO BE USED  
\$ITEMB: .BYTE 0,0 ;CONTAINS ITEM CONTROL BYTE  
;RESERVED--NOT TO BE USED  
\$HLTAD: .WORD 0 ;CONTAINS PC OF LAST HLT INSTRUCTION  
\$GDADR: .WORD 0 ;CONTAINS ADDRESS OF 'GOOD' DATA  
\$BDADR: .WORD 0 ;CONTAINS ADDRESS OF 'BAD' DATA  
\$GDAT: .WORD 0 ;CONTAINS 'GOOD' DATA  
\$BDAT: .WORD 0 ;CONTAINS 'BAD' DATA  
\$TMP0: .WORD 0 ;USER DEFINED  
\$TMP1: .WORD 0 ;USER DEFINED  
\$TMP2: .WORD 0 ;USER DEFINED  
TSTADR: .WORD 0 ;TEST ADDRESS  
LOLMT: .WORD 0 ;LOW LIMIT  
HILMT: .WORD 17776 ;HIGH LIMIT  
MOD: .WORD 0 ;MODULE NUMBER  
COL: .WORD 0 ;COLUMN NUMBER  
ROW: .WORD 0 ;ROW NUMBER  
PCHRG: .WORD 0 ;PRECHARGE  
CYCNT: .WORD 10000. ;CYCLE COUNT  
TSTPAT: .WORD -1 ;TEST PATTERN  
BCKGRD: .WORD -1 ;BACK GROUND PATTERN  
INSTAD: .WORD 0 ;ADDRESS OF INSTRUCTION UNDER TEST  
MSKBIT: .WORD BIT08 ;MASK BIT  
SV400: .WORD 0  
SVIN1: .WORD 0  
SVIN2: .WORD 0  
ESCAPE: .WORD 0  
\$BELL: .ASCIZ <207> ;CODE FOR BELL  
\$QUES: .ASCII /?/ ;QUESTION MARK  
\$CRLF: .ASCII <15> ;CARRIAGE RETURN  
\$LF: .ASCIZ <12> ;LINE FEED  
PASFLG: .WORD 0  
SELF: .WORD 0  
REDONE: .WORD -1  
LOBASE: .WORD 0  
HIBASE: .WORD 0  
LSTBK: .WORD 0  
PAGE: .WORD 0  
TO1: .WORD 0  
LODER: .WORD 0 ;RELOCATE TO HERE IF 0 CHOSEN  
;LOADER IS HERE

856  
857  
858  
859  
860  
861  
862  
863  
864  
865  
866  
867  
868  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
890

; THE FOLLOWING TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.  
; THIS 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 (\$HLTAD).  
; 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

EM1               ; TEST PATTERN  
DH1               ; ERROR FAILING  
                  ; PC    ADDRESS EXPECT   RCV'D   COL    ROW    PRECHARGE  
DT1               ; \$HLTAD \$BDADR \$GDDAT \$BDDAT \$TMPO \$TMP1 \$TMP2  
DF1               ; ALL NUMBERS ARE TYPED IN OCTAL

; ITEM 2

EM2               ; INSTRUCTION CHANGED  
DH2               ; ERROR INST    EXPECT   RCV'D,  
                  ; PC    ADDRESS DATA    DATA    COL    ROW    PRECHARGE  
DT1               ; \$HLTAD \$BDADR \$GDDAT \$BDDAT \$TMPO \$TMP1 \$TMP2  
DF1               ; ALL NUMBERS ARE TYPED IN OCTAL

; ITEM 3

EM3               ; BACK GROUND CHANGED  
DH3               ; ERROR BCKGRD EXPECT   RCV'D  
                  ; PC    ADDRESS DATA    DATA    COL    ROW    PRECHARGE  
DT1               ; \$HLTAD \$BDADR \$GDDAT \$BDDAT \$TMPO \$TMP1 \$TMP2  
DF1               ; ALL NUMBERS ARE TYPED IN OCTAL

\*\*\*\*\*

002674  
002674 007614  
002676 007707  
002700 010270  
002702 010310  
002704 007633  
002706 010016  
002710 010270  
002712 010310  
002714 007661  
002716 010143  
002720 010270  
002722 010310

891  
892  
893 002724 005010  
894 002726 077102  
895 002730 000207  
896  
897 002732 005010  
898 002734 077102  
899 002736 000207  
900  
901 002740 000000 177777 177777

\*\*\*\*\*  
; THIS IS WHERE THE TEST INSTRUCTIONS ARE STORED  
INSTR1: CLR (R0)  
INSTR2: SOB R1, INSTR1  
INSTR3: RTS PC  
; END OF TEST INSTRUCTIONS  
INSIDE: CLR (R0)  
SOB R1, INSIDE  
RTS PC  
OUTSID: .WORD 0, -1, -1

902  
 903  
 904  
 905  
 906  
 907  
 908  
 909  
 910  
 911 002746 010706  
 912 002750 062706 176130  
 913 002754 005067 177672  
 914  
 915 002760 004767 004254  
 916 002764 002647  
 917 002766 012767 000200 176326  
 918 002774 004767 176300  
 919 003000 004767 004234  
 920 003004 010771  
 921 003006 005767 176310  
 922 003012 100006  
 923 003014 016777 176746 177644  
 924 003022 012777 000001 175052  
 925 003030 016746 176730  
 926 003034 004067 004260  
 927 003040 006  
 928 003041 003  
 929 003042 005767 176254  
 930 003046 100002  
 931 003050 005077 175026  
 932 003054 004767 004160  
 933 003060 002647  
 934 003062 004767 004152  
 935  
 936 003066 010534  
 937 003070 012767 010653 177770  
 938 003076 004767 176672  
 939 003102 000000  
 940 003104 004767 004130  
 941 003110 010674  
 942 003112 012767 010753 177770  
 943 003120 004767 176650  
 944 003124 000000  
 945 003126 005067 177522  
 946 003132 012767 024042 002166  
 947 003140 012767 007736 002162  
 948 003146 012767 000100 002156  
 949 003154 012767 020100 002152  
 950  
 951  
 952  
 953  
 954  
 955  
 956  
 957

\*\*\*\*\*

; THIS IS THE STARTING POINT IF LIMITS ARE TO BE SELECTED  
 ; ALL TYPING SHOULD BE DONE ONLY AFTER SEEING SECTION  
 ; 4.3.1 AT THE BEGINNING OF THIS DOCUMENT

```

BEGIN1: MOV PC, SP ;SET STACK POINTER AND
        ADD #100-,SP ;MAKE IT POSITION INDEPENDANT
        CLR PASFLG ;PASFLG IS COUNTER FOR WHEN
                    ;BOTH PICKS & DROPS
        JSR PC, TYPE
        $CRLF
        MOV #BIT07,$SKT11 ;USE MEM MANG. IF THERE
        JSR PC, $SIZE ;SIZE MEM
        JSR PC, TYPE ;TYPE "LAST ADDRESS IN MEMORY"
        MEMEND
        TST $SKT11 ;IS MEMORY MANAGEMENT THERE?
        BPL 2$ ;BRANCH IF NO MEM MANAGEMENT
        MOV $LSTBK, $PAGE ;PAGE CONTAINS KIPAR6 OR KIPAR5
        MOV #1, $SR0 ;TURN ON MEM. MANAGEMENT
2$: MOV $LSTADR, -(SP) ;GET READY TO TYPE ADDRESS IN OCTAL
    JSR RD, $B20CT
    .BYTE 6
    .BYTE 3 ;PHYSICAL ADDRESS IF MEM MANG. THERE
    TST $SKT11
    BPL 3$
    CLR $SR0
3$: JSR PC, TYPE
    $CRLF
    JSR PC, TYPE ;GO TO TYPE "TYPE STARTING
                    ;BANK NUMBER AND CR
                    ;REFER TO SECTION 4.3.1 BEFORE CHOOSING"
1$: STARTB
    MOV #START2, 1$
    JSR PC, .RECD ;GET STARTING BANK NUMBER
                    ;AND LEAVE IT HERE
STBANK: .WORD 0
    JSR PC, TYPE ;GO TO TYPE "TYPE NUMBER OF 4K
                    ;BANKS TO BE TESTED AND CR"
1$: NUMBER
    MOV #NUMB2, 1$
    JSR PC, .RECD ;GET NUMBER OF BANKS
                    ;AND LEAVE IT HERE
BANKS: .WORD 0
        CLR SELF
        MOV #BEGIN2+20000, BEG2 ;THESE FOUR LINES
        MOV #7736, LENGTH ;TO SET UP FOR
        MOV #FIRST, FROM ;NO LIMIT CHOOSING ON RESTART
        MOV #FIRST+20000, TO
  
```

; THE FOLLOWING INSTRUCTIONS IN THIS SECTION IS FOR  
 ; RELOCATION IF STARTING BANK CHOSEN IS 0  
 ; IF RELOCATION IS ALREADY DONE BY OPERATOR THEN NO FURTHER  
 ; RELOCATION IS DONE

958	003162	016704	176600	MOV	\$LSTBK,R4	
959	003166	162704	300140	SUB	#140,R4	
960	003172	022704	001400	CMP	#1400,R4	
961	003176	101002		BHI	1\$	
962	003200	012704	001400	MOV	#1400,R4	
963	003204	000241		CLC		1\$:
964	003206	006104		ROL	R4	
965	003210	006104		ROL	R4	
966	003212	006104		ROL	R4	
967	003214	006104		ROL	R4	
968	003216	006104		ROL	R4	
969	003220	006104		ROL	R4	
970	003222	010402		MOV	R4,R2	
971	003224	005767	177442	TST	LODER	
972	003230	001027		BNE	5\$	;BRANCH IF SOMETHING IN LODER ;THAT IS LOADER ALREADY PULLED IN
973						
974	003232	010403		MOV	R4,R3	
975	003234	062703	020000	ADD	#20000,R3	
976	003240	010367	177426	MOV	R3,LODER	
977	003244	010705		MOV	PC,R5 ;TO	
978	003246	062705	006244	ADD	#LAST+2-.,R5	
979	003252	012700	003132	MOV	#17774-LAST/2,RO;LENGTH	
980	003256	014325		MOV	-(R3),(R5)+	2\$:
981	003260	077002		SOB	RO,2\$	
982	003262	004767	003752	JSR	PC,TYPE	
983	003266	011402		RELOO		
984	003270	012746	000324	MOV	#LOADER,-(SP)	
985	003274	004067	004020	JSR	RO,\$B2OCT	
986	003300	006		.BYTE	6	
987	003301	002		.BYTE	2	
988	003302	004767	003732	JSR	PC,TYPE	
989	003306	011434		RELOO2		
990	003310	005767	177566	TST	STBANK	5\$:
991	003314	001056		BNE	4\$	
992	003316	010746		MOV	PC,-(SP)	
993	003320	042726	017777	BIC	#17777,(SP)+	
994	003324	001052		BNE	4\$	
995	003326	010406		MOV	R4,SP	
996	003330	062706	001100	ADD	#1100,SP	
997	003334	062704	000100	ADD	#FIRST,R4	
998	003340	010467	177324	MOV	R4,T01	
999	003344	016700	001760	MOV	LENGTH,RO	
1000	003350	016701	001756	MOV	FROM,R1	
1001	003354	012124		MOV	(R1)+,(R4)+	3\$:
1002	003356	077002		SOB	RO,3\$	
1003	003360	062702	004042	ADD	#BEGIN2,R2	
1004	003364	004767	003650	JSR	PC,TYPE	
1005	003370	011264		PROTO		
1006	003372	016746	177272	MOV	T01,-(SP)	
1007	003376	062716	000104	ADD	#104,(SP)	
1008	003402	004067	003712	JSR	RO,\$B2OCT	
1009	003406	006		.BYTE	6	
1010	003407	002		.BYTE	2	
1011	003410	004767	003624	JSR	PC,TYPE	
1012	003414	002647		\$CRLF		
1013	003416	004767	003616	JSR	PC,TYPE	



1014	003422	011332			PROBAK		
1015	003424	016746	177240		MOV	T01, -(SP)	
1016	003430	062716	000200		ADD	#RESTOR-100, (SP)	
1017	003434	004067	003660		JSR	RO, \$B20CT	
1018	003440	006			.BYTE	6	
1019	003441	002			.BYTE	2	
1020	003442	004767	003572		JSR	PC, TYPE	
1021	003446	002647			\$CRLF		
1022	003450	000112			JMP	(R2)	
1023	003452	000573			BR	BEGIN2	
1024							
1025							
1026							
1027							
1028							
1029	003454	010706			BEGN11: MOV	PC, SP	; SETUP STACK POINTER AND
1030	003456	062706	175422		ADD	#1100-, SP	; MAKE IT POSITION INDEPENDANT
1031	003462	005067	177164		CLR	PASFLG	; PASFLG IS COUNTER WHEN BOTH PICKS
1032							; AND DROPS ARE USED
1033	003466	012767	000200	175626	MOV	#BIT07, \$KT11	; BIT 7 SAYS SEGMENTATION IS THERE
1034	003474	004767	175600		JSR	PC, \$SIZE	; SIZE MEMORY
1035	003500	004767	003534		JSR	PC, TYPE	
1036	003504	010771			MEMEND		
1037	003506	005767	175610		TST	\$KT11	
1038	003512	100006			BPL	2\$	
1039	003514	016777	176246	177144	MOV	\$LSTBK, @PAGE	
1040	003522	012777	000001	174352	MOV	#1, @SRO	
1041	003530	016746	176230		2\$: MOV	\$LSTADR, -(SP)	
1042	003534	004067	003560		JSR	RO, \$B20CT	
1043	003540	006			.BYTE	6	
1044	003541	003			.BYTE	3	
1045	003542	005767	175554		TST	\$KT11	
1046	003546	100002			BPL	3\$	
1047	003550	005077	174326		CLR	@SRO	
1048	003554	004767	003460		3\$: JSR	PC, TYPE	
1049	003560	002647			\$CRLF		
1050	003562	012767	000001	177312	MOV	#1, STBANK	; STARTING BANK # = 1 LOC = 20000
1051	003570	016767	176172	177066	MOV	\$LSTBK, LSTBK	; LAST BANK IS STORED IN LSTBK
1052	003576	062767	000040	177060	ADD	#40, LSTBK	
1053	003604	006067	177054		ROR	LSTBK	; DIVIDE LSTBK BY 200
1054	003610	006067	177050		ROR	LSTBK	; TO GIVE NUMBER OF
1055	003614	006067	177044		ROR	LSTBK	; BANKS PRESENT IN
1056	003620	006067	177040		ROR	LSTBK	; EXISTING MEMORY
1057	003624	006067	177034		ROR	LSTBK	
1058	003630	006067	177030		ROR	LSTBK	
1059	003634	006067	177024		ROR	LSTBK	
1060	003640	005367	177020		DEC	LSTBK	; NOW DECREASE ONE FOR BANKS
1061	003644	016767	177014	177252	MOV	LSTBK, BANKS	; TO BE TESTED AND STORE
1062	003652	012767	177777	176774	MOV	#-1, SELF	; 1 = RELOCATION REQUIRED
1063	003660	012767	024042	001440	MOV	#BEGIN2+20000, BEG2	
1064	003666	012767	007736	001434	MOV	#7736, LENGTH	
1065	003674	012767	000100	001430	MOV	#FIRST, FROM	
1066	003702	012767	020100	001424	MOV	#FIRST+20000, TO	
1067	003710	005067	176742		CLR	REDONE	; RELOCATION EXIT COUNTER
1068	003714	005767	176752		TST	LODER	
1069	003720	001050			BNE	BEGIN2	; BRANCH IF SOMETHING IN LODER

```

1070
1071 003722 016704 176040
1072 003726 162704 000140
1073 003732 022704 001400
1074 003736 101002
1075 003740 012704 001400
1076 003744 000241
1077 003746 006104
1078 003750 006104
1079 003752 006104
1080 003754 006104
1081 003756 006104
1082 003760 006104
1083 003762 010402
1084 003764 010403
1085 003766 062703 020000
1086 003772 010367 176674
1087 003776 010705
1088 004000 062705 005512
1089 004004 012700 003132
1090 004010 014325
1091 004012 077002
1092 004014 004767 003220
1093 004020 011402
1094 004022 012746 000324
1095 004026 004067 003266
1096 004032 006
1097 004033 002
1098 004034 004767 003200
1099 004040 011434
1100
1101
1102
1103
1104 004042 004767 003172
1105 004046 002647
1106 004050 004767 003164
1107 004054 010350
1108 004056 012767 010515 177770
1109 004064 000000
1110 004066 032737 040000 177570
1111 004074 001402
1112 004076 004767 176074
1113
1114
1115
1116
1117
1118 004102 016700 176774
1119 004106 005200
1120 004110 012703 177600
1121 004114 016701 177004
1122 004120 012702 160000
1123 004124 062702 020000
1124 004130 062703 000200
1125 004134 077005

```

;THAT IS LOADER ALREADY PULLED IN

1\$:

4\$:

;THE CONSOLE SWITCHES ARE TO BE SET. LOOK AT SECTION  
;4.1 AT THE FRONT OF THIS DOC

BEGIN2:

1\$:

;CONVERT THE TYPED LIMITS INTO HIGH AND LOW LIMITS  
;IF SELF STARTING USED THEN "STBANK" AND "BANKS"  
;HAVE ALREADY BEEN FILLED FOR USE HERE

BEGIN3:

1\$:

```

MOV $LSTBK,R4
SUB #140,R4
CMP #1400,R4
BHI 1$
MOV #1400,R4
CLC
ROL R4
ROL R4
ROL R4
ROL R4
ROL R4
MOV R4,R2
MOV R4,R3
ADD #20000,R3
MOV R3,LODER
MOV PC,R5 ;TO
ADD #LAST+2-,R5
MOV #17774-LAST/2,R0;LENGTH
SOB R0,4$
JSR PC,TYPE
RELOD
MOV #LOADER,-(SP)
JSR R0,$B20CT
.BYTE 6
.BYTE 2
JSR PC,TYPE
RELOD2

```

;GO TO TYPE "SET SWITHCHES  
;FOR OPERATNG MODE IE. PICKUPS  
;OR DROPS OR BOTH THEN HIT CONTINUE"  
;PARITY CHECKING BIT SET?  
;IF NOT BRANCH  
;GO TO ENABLE PARITY

;MAKE WORKING LOCATION FOR LOW LIMIT  
;R3 WILL HAVE PAGE ADDRESS  
;MAKE WORKING LOCATION FOR HIGH LIMIT  
;R2 WILL HAVE LOW LIMIT  
;GETTING LOW LIMIT  
;GETTING LOW LIMIT PAGE ADDRESS  
;REPEAT R0 TIMES

```

1126 004136 010367 176516      MOV      R3,      LOBASE  ;STORE LOW LIMIT PAGE ADDRESS
1127 004142 010267 176440      MOV      R2,      LCLMT   ;SET LOW LIMIT
1128 004146 010267 176440      MOV      R2,      MOD     ;SET MODULE NUMBER
1129 004152 162703 000200      SUB      #200,    R3      ;GETTING READY FOR HIGH LIMIT PAGE
1130 004156 062702 020000      BEGIN4: ADD     #20000,  R2     ;GETTING HIGH LIMIT
1131 004162 062703 000200      ADD      #200,    R3      ;GETTING HIGH LIMIT PAGE ADDRESS
1132 004166 077105          SOB      R1,      BEGIN4  ;REPEAT FOR BANK NUMBER
1133 004170 005302          DEC      R2,      ;GETTING HIGH LIMIT WITHIN BANK
1134 004172 010267 176412      MOV      R2,      HILMT   ;STORE HIGH LIMIT
1135 004176 010367 176460      MOV      R3,      HIBASE  ;STORE HIGH LIMIT PAGE ADDRESS
1136 004202 022767 000006 176672      CMP      #6,      STBANK  ;DOES STARTING BANK NEED SEGMENTATION?
1137 004210 103401          BLO     1$,      ;BRANCH IF YES
1138 004212 000407          BR      2$,      ;
1139 004214 162767 000200 176436 1$: SUB      #200,    LOBASE  ;SEGMENTATION REQUIREG 50
1140 004222 005167 176424          COM     PASFLG
1141 004226 000167 000716          JMP     SEGMT1
1142 004232 022703 001600      2$: CMP      #1600,  R3    ;SUBTRACT TO ENABLE REPEATED ADDITION
1143 004236 101003          BHI     BEGINS
1144 004240 012767 157776 176342      MOV      #157776, HILMT  ;IS SEGMENTATION REQUIRED FOR HIGH?
1145          ;IF NO THEN BRANCH
1146          ;IF YES HIGH LIMIT = END OF 32K
1147          ;SETUP ALL TAGS TO THEIR APPROPRIATE VALUES
1148 004246 005067 176342      BEGINS: CLR     COL        ;MAKE COLUMN ZERO
1149 004252 005067 176340      CLR     ROW        ;MAKE ROW ZERO
1150 004256 005067 176336      CLR     PCHRG      ;MAKE PRECHARGE ZERO
1151 004262 012767 023420 176332      MOV      #10000.,  CYCNT  ;SETUP CYCLE COUNT
1152 004270 012767 177777 176326      MOV      #-1,     TSTPAT ;SETUP TEST PATTERN
1153 004276 012767 177777 176322      MOV      #-1,     BCKGRD ;SETUP BACKGROUND
1154 004304 012767 000400 176320      MOV      #BIT08,  MSKBIT ;SETUP MASK
1155
1156
1157
1158
1159 004312 032737 010000 177570      BEGIN6: BIT     #SW12, #SWR  ;IS SWITCH 12 SET?
1160 004320 001013          BNE     1$,      ;IF SET BRANCH TO CHECK SW11
1161 004322 005167 176324          COM     PASFLG      ;COMPLEMENT PASS FLAG
1162          ;IS BOTH PICKUPS AND DROPS DONE
1163 004326 001037          BNE     3$,      ;IF YES BRANCH
1164 004330 005067 176272      CLR     BCKGRD    ;IF NOT SETUP TO
1165 004334 005067 176264      CLR     TSTPAT    ;CATCH
1166 004340 012767 177777 176372      MOV      #-1,     OUTSID ;BIT PICKS
1167 004346 000437          BR      4$,      ;
1168 004350 005067 176276      1$: CLR     PASFLG
1169 004354 032737 010000 177570      BIT     #SW12, #SWR  ;TEST SW12
1170 004362 001410          BEQ     2$,      ;IF NOT SET BRANCH TO CATCH DROPS
1171 004364 005067 176236      CLR     BCKGRD    ;IF NOT SET TO CATCH PICKUPS
1172 004370 005067 176230      CLR     TSTPAT
1173 004374 012767 177777 176336      MOV      #-1, OUTSID
1174 004402 000421          BR      4$,      ;
1175 004404 012767 177777 176212      2$: MOV      #-1,     TSTPAT
1176 004412 012767 177777 176206      MOV      #-1,     BCKGRD
1177 004420 005067 176314      CLR     OUTSID
1178 004424 000410          BR      4$,      ;
1179 004426 012767 177777 176170      3$: MOV      #-1,     TSTPAT
1180 004434 012767 177777 176164      MOV      #-1,     BCKGRD
1181 004442 005067 176272      CLR     OUTSID

```

```

1182 004446 005067 176066      45:  CLR      $PASS      ;CLEAR PASS COUNT
1183
1184
1185
1186
1187      ;MAKE THE TEST ADDRESS AND THE INSTRUCTION ADDRESS
1188
1189
1190
1191
1192 004452 016746 176134      LOOP1:  MOV      MOD,      -(SP)      ;PICKUP MODULE NUMBER
1193 004456 042716 017777      BIC      #17777,(SP)      ;STRIP AWAY ANY JUNK
1194 004462 016746 176126      MOV      COL,      -(SP)      ;PICKUP COLUMN NUMBER
1195 004466 042716 160777      BIC      #17000,(SP)      ;STRIP AWAY JUNK
1196 004472 032767 020000      BIT      #BIT13,COL      ;CHECK C4=1
1197 004500 001402      BEQ      15              ;BR IF NO
1198 004502 052716 000002      BIS      #BIT01,(SP)      ;PUT C4 AT PROPER PLACE IN BUS ADDRESS
1199 004506 016746 176106      15:     MOV      PCHRG,      -(SP)      ;PICK UP PRECHARGE
1200 004512 042716 177763      BIC      #14,(SP)      ;STRIP JUNK
1201 004516 016746 176074      MOV      ROW,      -(SP)      ;GET THE ROW NUMBER
1202 004522 042716 177017      BIC      #1760,(SP)      ;STRIP THE JUNK IF ANY
1203 004526 012600      MOV      (SP)+,RO      ;FORM THE BUS ADDRESS WITH ROW
1204 004530 052600      BIS      (SP)+,RO      ;PRECHARGE
1205 004532 052600      BIS      (SP)+,RO      ;COLUMN
1206 004534 052600      BIS      (SP)+,RO      ;MODULE
1207 004536 010067 176042      MOV      RO,INSTADR      ;SAVE AS THE TEST ADDRESS
1208 004542 010067 176062      MOV      RO,INSTAD      ;AND AS THE INSTRUCTION ADDRESS
1209
1210
1211
1212      ;MOVE EITHER "0 -1 -1" OR THE "CLEAR RO"
1213      ;INTO THE LOCATION INSTR1: THIS WILL FINALLY BE THE TEST INSTRUCTION
1214
1215
1216
1217 004546 010700      LPERR:  MOV      PC,      RO      ;THESE TWO INSTRUCTIONS
1218      ADD      #INSTR1-.,RO      ;GETS THE LOCATION
1219 004550 062700 176154      ADD      #INSTR1-.,RO      ;OF INSTR1 IN TO RO
1220      MOV      #SWR, SWITCH      ;EVEN AFTER RELOCATION
1221 004554 013767 177570 000160      MOV      #SWR, SWITCH      ;GET SWITCH SETTING
1222      BIC      #177577,      ;TO CHECK FOR
1223 004562 042767 177577 000152      BIC      #177577,      SWITCH ;IF SW07=0 DO OUTSIDE
1224
1225      ;IF SW07=1 DO INSIDE
1226
1227 004570 001404      BEQ      15              ;IF SW07=0 THEN BRANCH
1228
1229 004572 010701      MOV      PC,      R1      ;SW07=1 SO SETUP TO
1230      ADD      #INSIDE-.,R1      ;MOVE INSIDE INTO INSTR1
1231 004574 062701 176136      ADD      #INSIDE-.,R1      ;MAKE INSIDE POSITION
1232      BR      25              ;INDEPENDANT
1233 004600 000403      BR      25
1234
1235 004602 010701      15:     MOV      PC,      R1      ;SW07=0 SO SETUP TO
1236 004604 062701 176134      ADD      #OUTSID-.,R1      ;MOVE OUTSIDE INTO INSTR1
1237      ;MAKE OUTSIDE POSITION

```

```

1238 004610 012120 25: MOV (R1)+, (R0)+ ; INDEPENDANT
1239 ; THESE THREE INSTRUCTIONS
1240 004612 012120 MOV (R1)+, (R0)+ ; MOVES THE APPROPRIATE
1241 ; GROUP INTO INSTR1
1242 004614 012120 MOV (R1)+, (R0)+ ; GROUP INTO INSTR1
1243 004616 105067 175732 CLRB $ERFLG ; CLEAR ERROR FLAG
1244
1245 ;FILL 4K WITH BACKGROUND PATTERN
1246
1247 004622 016700 176000 FILL4K:MOV BCKGRD, R0 ; GETTING READY TO MOVE
1248 ; BACKGROUND INTO 4K
1249 004626 016701 175760 MOV MOD, R1 ; STARTING FROM MODULE
1250 ; NUMBER
1251 004632 012702 010000 15: MOV #10000, R2 ; 10000=4K
1252 004636 010021 MOV R0, (R1)+ ; FILL BACKGROUND
1253 004640 077202 SOB R2, 15 ; REPEAT 4K TIMES
1254
1255 ; THIS MOVES CONTENTS OF TEST PATTERN (TSTPAT)
1256 ; HERE = -1 INTO MOD AND ALL LOCATIONS GOT
1257 ; BY INCREASING COLUMN BY ONE TILL ONE 4K
1258 ; IS COMPLETE
1259 LDPAT: MOV TSTADR, R0 ; MOVE TEST ADDRESS
1260 MOV R0, R1 ; DUPLICATE IT IN R1 TO BE REINSTATED AFTER ADDING COLUMN
1261 BIC #C160774, R1 ; DON'T KEEP THE COL
1262 MOV COL, R2 ; PICK UP THE COL
1263 MOV #BIT13:BIT01, R3 ; GET READY TO TRANSFER BIT 13 TO BIT 1
1264 MOV TSTPAT, R4 ; MOVE TEST PATTERN=-1
1265 15: ADD #1000, R2 ; ADD ONE TO COLUMN
1266 BIC #BIT14, R2 ; IF IT WAS ALREADY 37 THEN ADDING ONE WILL GIVE BIT 14
1267 MOV R2, R5 ; FORM AN ADDRESS
1268 BIT #BIT13, R5 ; POSITION C4 IF NEEDED
1269 BEQ 25 ; IF BIT 13 IS ZERO THEN BRANCH
1270 XOR R3, R5 ; IF BIT 13=1 THEN TRANSFER TO BIT 1
1271 25: BIS R1, R5 ; REINSTATE OTHER BITS AFTER ADDING TO COLUMN
1272 MOV R4, (R5) ; STORE THE PATTERN
1273 CMP R0, R5 ; TSTADR?
1274 BNE 15 ; BR IF NO
1275
1276 ; THE FOLLOWING JSR LOADS INSRTUCTION (INSTR1) INTO
1277 ; PROPER PLACE
1278
1279 LOOP2: JSR PC, LDINT
1280
1281 BR NEXT
1282
1283 004722 004767 000732
1284 004726 000431

```

```

1294
1295 004730 011067 175700          MOV      (R0), SV400
1296
1297 004734 016701 175662          MOV      CYCNT, R1      ;DO TEST SEQ. THIS
1298                                ;MANY TIMES
1299 004740 005727          TST      (PC)+
1300 004742 000000          SWITCH: .WORD 0
1301 004744 001406          BEQ      1$
1302 004746 016746 175656          MOV      INSTAD, -(SP)  ;SETUP THE STARTING ADDRESS
1303 004752 011637 177570          MOV      (SP), 2#DISPLAY ;DISPLAY THE TEST ADDRESS
1304 004756 004736          JSR      PC, 2(SP)+     ;GO TO THE TEST SEQ.
1305 004760 000406          BR       3$
1306 004762 016702 175642          1$:     MOV      INSTAD, R2
1307 004766 010237 177570          MOV      R2, 2#DISPLAY
1308 004772 011210          2$:     MOV      (R2), (R0)
1309 004774 077102          SOB      R1, 2$
1310 004776 016710 175632          3$:     MOV      SV400, (R0)
1311
1312
1313                                ;CHECK THE 1K BLOCK
1314
1315
1316 005002 004767 001074          JSR      PC, CHECK      ;GO CHECK THE 1K BLOCK
1317 005006 000401          BR      NEXT           ;RETURN HERE ON ERROR
1318 005010 000744          BR      LOOP2
1319 005012 032737 000001 177570  NEXT:   BIT      #BIT00, 2#SWR  ;LOCK ON THIS PRECHARGE?
1320 005020 001007          BNE     1$            ;BR IF YES
1321 005022 062767 000004 175570          ADD     #4, PCHRG      ;GO TO NEXT PRECHARGE
1322 005030 042767 177763 175562          BIC     #1C14, PCHRG  ;GET RID OF TRASH
1323 005036 001205          BNE     LOOP1         ;LOOP IF MORE TO DO
1324 005040 032737 000002 177570  1$:     BIT      #BIT01, 2#SWR  ;LOCK ON THIS MODULE?
1325 005046 001133          BNE     SEOP          ;BR IF YES
1326 005050 062767 020000 175534          ADD     #BIT13, MOD   ;INCREMENT THE MOD
1327 005056 103407          BCS     SEGMT        ;IF ADDING BIT13 MAKES 0
1328                                ;THEN BRANCH
1329 005060 026767 175526 175522          CMP     MOD, HILMT   ;DON'T LET IT GET TO BIG
1330 005066 103401          BLO     LOOP12
1331 005070 000402          BR      SEGMT
1332 005072 000167 177354          LOOP12: JMP     LOOP1  ;GO TO CHECK IF SEGMENTATION IS REQUIRED
1333                                ;RETURN
1334                                ;IS SEGMENTATION REQUIRED? IF SO TURN ON MEMORY
1335                                ;MANAGEMENT AND SETUP ALL REQUIRED REGISTERS
1336                                ;ONLY KERNEL MODE INSTRUCTION SPACE IS USED
1337 005076 022767 001600 175556          SEGMT:  CMP     #1600, HIBASE ;SEGMENTATION REQUIRED FOR HIGH LIMIT?
1338 005104 101114          BHI     SEOP          ;IF NO THEN BRANCH
1339 005106 022767 001600 175544          CMP     #1600, LOBASE ;IF YES THEN TEST IF LOW LIMIT REQ SEGMENTATION?
1340 005114 101415          BLOS   SEGMT1        ;IF LOW LIMIT DOES REQ SEGMENTATION BRANCH
1341 005116 012767 001600 175534          MOV     #1600, LOBASE ;IF NO MAKE LOW LIMIT END OF 32K
1342 005124 012767 140000 175454          SEGO:  MOV     #140000, LOLMT ;LOLMT LOOKS AT PAGE OR 6
1343 005132 012767 140000 175452          MOV     #140000, MOD  ;SAME WITH MODULE NUMBER
1344 005140 012767 157776 175442          MOV     #157776, HILMT ;HIGH LIMIT LOOKS AT PAGE END OF 4K
1345 005146 000420          BR      SEGMT2        ;BRANCH TO ENABLE SEGMENTATION
1346 005150 062767 000200 175502          SEGMT1: ADD    #200, LOBASE ;DO NEXT 4K
1347 005156 012767 140000 175422          SEG10: MOV     #140000, LOLMT ;MAKE LOW LIMIT AND MODULE
1348 005164 012767 140000 175420          MOV     #140000, MOD  ;0 ON PAGE "PAGE OR 6"
1349 005172 012767 157776 175410          MOV     #157776, HILMT ;HIGH LIMIT SAME 4K ON PAGE

```

DCMS88.P11

1350	005200	026767	175456	175452	CMP	HIBASE, LOBASE	; IS IT THE END?
1351	005206	103453			BLO	\$EOP	; IF YES BRANCH
1352	005210	016777	175444	175450	SEGMT2: MOV	LOBASE, 2PAGE	; SET LOW LIMIT ON PAGE 4
1353	005216	005077	172666		CLR	2SR3	; USE ONLY I SPACE IN KERNAL
1354	005222	012700	077406		MOV	#77406, RO	; MOVE INTO RO
1355	005226	010077	172660		MOV	RO, 2KIPDR0	; SEGMENT DESCRIPTOR
1356	005232	010077	172656		MOV	RO, 2KIPDR1	; REGISTER IS SET FOR
1357	005236	010077	172654		MOV	RO, 2KIPDR2	; 128 BLOCKS WITH NO
1358	005242	010077	172652		MOV	RO, 2KIPDR3	; SYSTEM TRAPS. THESE
1359	005246	010077	172650		MOV	RO, 2KIPDR4	; REGISTERS ARE ALSO
1360	005252	010077	172646		MOV	RO, 2KIPDR5	; CALLED PAGE
1361	005256	010077	172644		MOV	RO, 2KIPDR6	; DESCRIPTOR REGISTERS
1362	005262	010077	172642		MOV	RO, 2KIPDR7	; ONLY PAGE 0 AND 4 IS USED
1363	005266	005077	172640		CLR	2KIPDR0	; MAKE PAGE ADDRESS 0=0
1364	005272	012777	000001	172602	MOV	#1, 2SR0	; ENABLE SEGMENTATION
1365	005300	032737	010000	177570	BIT	#SW12, 2SWR	; TEST SWITCH 12
1366	005306	000003			BNE	2\$	; BRANCH IF SET
1367	005310	005167	175336		COM	PASFLG	; SW12 NOT SET IE. DO BOTH
1368	005314	000402			BR	3\$	; SO COMPLEMENT AND RESTART
1369	005316	005067	175330	2\$:	CLR	PASFLG	; SW12 SET IE. DO ONE SO RESTART
1370	005322	000167	176764	3\$:	JMP	BEGIN6	

1371  
1372  
1373  
1374  
1375  
1376  
1377  
1378  
1379  
1380  
1381  
1382  
1383  
1384  
1385  
1386  
1387

```

:*****
:END OF PASS ROUTINE
:INCREMENT THE PASS NUMBER
:IF SW10=0 PRINT END PASS ON END OF PROGRAM
:IF THERE IS A MONITOR GO TO IT.
:IF NONE JUMP TO BEGIN3
:IF LIMITS ARE NOT CHOSEN AND ALL OF MEMORY
:IS TO BE TESTED THIS RELOCATES PROGRAM FROM
:FIRST 4K TO SECOND 4K AND SETS UP FOR
:TESTING THE FIRST 4K. THEN RELOCATES
:TO FIRST 4K TO BE ABLE TO START AGAIN.
    
```

1388	005326	024042			BEG2:	20000+BEGIN2
1389	005330	007736			LENGTH:	7736
1390	005332	000100			FROM:	FIRST
1391	005334	020100			TO:	20000 + FIRST

1392  
1393  
1394

1395	005336	005767	175312		\$EOP:	TST	SELF	; WAS SELF STARTING USED
1396	005342	001523				BEQ	\$EOP1	; IF NOT IE. LIMITS WERE CHOSEN THEN BRANCH
1397								; HAS ENTRY INTO THIS SECTION BEEN MADE?
1398								; IF YES IE. THIS IS SECOND TIME THEN BRANCH

1399  
1400  
1401  
1402  
1403  
1404  
1405

1399	005344	005767	173752		TST	\$KT11	
1400	005350	100005			BPL	5\$	
1401	005352	005077	172524		CLR	2SR0	
1402	005356	000240			NOP		
1403	005360	000240			NOP		
1404	005362	000240			NOP		
1405	005364	016700	177740	5\$:	MOV	LENGTH, RO	; IF NOT IE. FIRST TIME OR THIRD TIME

1406	005370	016701	177736			MOV	FROM, R1	:GETTING READY TO RELOCATE
1407	005374	016702	177734			MOV	TO, R2	:GETTING READY TO RELOCATE
1408	005400	012767	000000	175474		MOV	#0, STBANK	:FILL IN TYPED INFORMATION
1409	005406	012767	000001	175510		MOV	#1, BANKS	:ABOUT STARTING ADDRESSES
1410	005414	005067	175166			CLR	L0LMT	:SETUP LIMITS
1411	005420	005067	175166			CLR	MOD	:MODULE
1412	005424	012767	017775	175156		MOV	#17776, HILMT	:AND HIGH
1413	005432	012122			15:	MOV	(R1)+, (R2)+	:FROM INTO TO
1414	005434	077002				SOB	R0 15	:REPEAT
1415	005436	005167	175214			COM	REDONE	:BEEN THROUGH ONCE?
1416	005442	016705	175210			MOV	REDONE, R5	:
1417	005446	001401				BEG	25	:BRANCH IF SECOND TIME THROUGH
1418	005450	000433				BR	35	:FIRST TIME GO TO 35
1419	005452	012737	000001	003102	25:	MOV	#1, @#STBANK	
1420	005460	016737	175200	003124		MOV	LSTBK, @#BANKS	
1421	005466	012706	001100			MOV	#1100, SP	
1422	005472	000240				NOP		
1423	005474	010537	002656			MOV	R5, @#REDONE	
1424	005500	012737	000100	005332		MOV	#FIRST, @#FROM	
1425	005506	012737	020100	005334		MOV	#20000+FIRST, @#TO	
1426	005514	000240				NOP		
1427	005516	000240				NOP		
1428	005520	000240				NOP		
1429	005522	000240				NOP		
1430	005524	000240				NOP		
1431	005526	000240				NOP		
1432	005530	000137	005612			JMP	@#SEOP1	
1433	005534	000137	004102		45:	JMP	@#BEGIN3	
1434	005540				35:			
1435	005540	012706	021100			MOV	#21100, SP	
1436	005544	010537	022656			MOV	R5, @#20000+REDONE	
1437	005550	012701	005332			MOV	#FROM, R1	:
1438	005554	012702	005334			MOV	#TO, R2	:
1439	005560	062701	020000			ADD	#20000, R1	:
1440	005564	062702	020000			ADD	#20000, R2	:
1441	005570	016703	177536			MOV	FROM, R3	:
1442	005574	016711	177534			MOV	TO, (R1)	:
1443	005600	010312				MOV	R3, (R2)	:
1444	005602	005137	022652			COM	@#20000+PASFLG	:
1445	005606	000137	024312			JMP	@#BEGIN6+20000	:
1446								
1447	005612	005767	175034		SEOP1:	TST	PASFLG	:
1448	005616	001011				BNE	SEOP3	:
1449	005620	005267	174714		SEOP2:	INC	\$PASS	:INCREMENT PASS COUNT
1450	005624	032737	002000	177570		BIT	#SW10, @#SWR	:TYPE END PASS?
1451	005632	001003				BNE	SEOP3	:
1452	005634	004767	001400			JSR	PC, TYPE	:
1453	005640	010335				EPASS		:
1454	005642	005767	173454		SEOP3:	TST	\$KT11	:
1455	005646	100002				BPL	15	:
1456	005650	005077	172226			CLR	@#SRO	:
1457	005654	000167	176222		15:	JMP	BEGIN3	:
1458								
1459								
1460								
1461								



1462	005660					LDINT:			
1463	005660	010246				MOV	R2, -(SP,		; PUSH R2 ON STACK
1464	005662	010346				MOV	R3, -(SP)		; PUSH R3 ON STACK
1465	005664	011700	174740		15:	MOV	INSTAD, R0		; GET THE INSTRUCTION ADDRESS
1466	005670	042700	177017			BIC	#1C760, R0		; KEEP ROW
1467	005674	020067	174716			CMP	R0, ROW		; AT THE TEST ROW?
1468	005700	071412				BEQ	4\$		; BR IF YES
1469	005702	016746	174722			MOV	INSTAD, -(SP)		
1470	005706	016776	174714	000000		MOV	BCKGRD, 2(SP)		; RESTORE BACK GROUND
1471	005714	062716	000002			ADD	#2, (SP)		
1472	005720	016736	174714			MOV	SVIN2, 2(SP)+		
1473	005724	000403				BR	5\$		
1474	005726	016777	174672	174674	4\$:	MOV	TSTPAT, 2INSTAD		
1475	005734	016777	174664	174642	5\$:	MOV	TSTPAT, 2TSTADR		; STORE THE TEST PATTERN
1476	005742	016700	174662			MOV	INSTAD, R0		; GET THE INSTRUCTIONS ADDRESS
1477	005746	004767	000600		2\$:	JSR	PC, FORMAD		; GO FORM NEXT INSTUCTION ADDRESS
1478	005757	010003				MOV	R0, R3		; CHECK FOR OUT OF RANGE
1479	005754	016702	174624			MOV	TSTADR, R2		; PICKUP THE TSTADR FOR CHECKING
1480	005760	020302				CMP	R3, R2		; IS INSTR. ADDRESS = TEST ADDRESS?
1481	005762	001443				BEQ	3\$		; BR IF YES
1482	005764	062703	000002			ADD	#2, R3		; ADDRESS WHERE INSTR2 GOES
1483	005770	020302				CMP	R3, R2		; IS IT THE SAME AS TSTADR
1484	005772	001437				BEQ	3\$		; BR IF YES
1485	005774	062703	000002			ADD	#2, R3		; INSTR3'S ADDRESS
1486	006000	020302				CMP	R3, R2		; SAME AS TSTADR?
1487	006002	001433				BEQ	3\$		; BR IF YES
1488	006004	020367	174600			CMP	R3, HILMT		; WILL IT GO OFF THE END?
1489	006010	101356				BHI	2\$		; BR IF YES
1490	006012	010067	174612			MOV	R0, INSTAD		; SAVE NEW ADDRESS
1491	006016	011067	174614			MOV	(R0), SVIN1		; SAVE WHATS AT THE 1ST TWO
1492	006022	016067	000002	174610		MOV	2(R0), SVIN2		; TEST INSTRUCTIONS ADDRESSES
1493	006030	016710	174670			MOV	INSTR1, (R0)		; LOAD THE INSTRUCTIONS
1494	006034	016760	174666	000002		MOV	INSTR2, 2(R0)		
1495	006042	016760	174662	000004		MOV	INSTR3, 4(R0)		
1496	006050	016746	174556			MOV	MSKBIT, -(SP)		; PICKUP THE MASK BIT
1497	006054	074016				XOR	R0, (SP)		; HALF ADD TO THE ADDRESS
1498	006056	012600				MOV	(SP)+, R0		; GET THE ADDRESS TO BE WRITTEN INTO
1499	006060	020002				CMP	R0, R2		; IS IT THE SAME AS THE TEST ADDRESS?
1500	006062	001700				BEQ	1\$		; BR IF YES
1501	006064	062766	000002	000004		ADD	#2, 4(SP)		; ADJUST THE RETURN ADDRESS
1502	006072				3\$:				
1503	006072	012603				MOV	(SP)+, R3		; POP STACK INTO R3
1504	006074	012602				MOV	(SP)+, R2		; POP STACK INTO R2
1505	006076	000207				RTS	PC		; GO BACK
1506									
1507									
1508									
1509	006100	000000				TCOL:	0		
1510									
1511	006102	010246				CHECK:	MOV	R2, -(SP)	; PUSH R2 ON STACK
1512	006104	010346					MOV	R3, -(SP)	; PUSH R3 ON STACK
1513	006106	010446					MOV	R4, -(SP)	; PUSH R4 ON STACK
1514	006110	010546					MOV	R5, -(SP)	; PUSH R5 ON STACK
1515	006112	016701	174466			MOV	TSTADR, R1		
1516	006116	016702	174506			MOV	INSTAD, R2		; ADDRESS OF THE TEST INSTR1
1517	006122	010200				MOV	R2, R0		

1518	006124	010203		MOV	R2,R3	
1519	006126	062703	000002	ADD	#2,R3	; ADDRESS OF THE TEST INSTR2
1520	006132	010004		MOV	RO,R4	; USED TO KEEP TRACK OF THE TEST ROW
1521	006134	042704	177017	BIC	#1C760,R4	
1522	006140	010067	177734	MOV	RO,TCOL	; USED TO KEEP TRACK OF THE TEST COL
1523	006144	042767	160775	BIC	#1C17002,TCOL	
1524	006152	010746		MOV	PC,-(SP)	; SETUP THE ESCAPE ON ERROR ADDRESS
1525	006154	062716	000352	ADD	#CHK1-(SP)	; MAKE IT POSITION INDEPENDANT
1526	006160	012667	174456	MOV	(SP)+,ESCAPE	
1527	006164	000407		BR	1\$	
1528						
1529	006166	042700	017762	8\$: BIC	#17762,RO	; FORM ADDRESS
1530	006172	050400		BIS	R4,RO	; GET THE ROW
1531	006174	056700	177700	BIS	TCOL,RO	; AND THE COLUMN
1532	006200	020002		CMP	RO,R2	; IS THIS THE INSTRUCTION ADDRESS?
1533	006202	001551		BEQ	CHK1	; BR IF YES
1534						
1535	006204	011005		1\$: MOV	(RO),R5	; PICK UP THE DATA
1536	006206	020001		CMP	RO,R1	; AT TSTADR?
1537	006210	001016		BNE	2\$	; BR IF NO
1538						
1539	006212	020567	174406	CMP	R5,TSTPAT	; DID THE TSTPAT UNDER TEST CHANGE?
1540	006216	001515		BEQ	7\$	; BR IF NO
1541	006220	010567	174350	MOV	R5,\$BDDAT	
1542	006224	016767	174374	MOV	TSTPAT,\$GDDAT	174340
1543	006232	010067	174332	MOV	RO,\$BDADR	
1544	006236	004767	000352	JSR	PC,\$HLT	
1545	006242	000001		1		
1546	006244	000502		BR	7\$	
1547						
1548	006246	020002		2\$: CMP	RO,R2	; AT INSTAD
1549	006250	001020		BNE	4\$	; BR IF NO
1550	006252	020567	174446	CMP	R5,INSTR1	; DID INSTR1 UNDER TEST CHANGE?
1551	006256	001412		BEQ	3\$	; BR IF NO
1552	006260	010567	174310	MOV	R5,\$BDDAT	
1553	006264	016767	174434	MOV	INSTR1,\$GDDAT	174300
1554	006272	010067	174272	MOV	RO,\$BDADR	
1555	006276	004767	000312	JSR	PC,\$HLT	
1556	006302	000002		2		
1557	006304	016710	174326	3\$: MOV	SVIN1,(RO)	; RESTORE THIS LOCATION
1558	006310	000460		BR	7\$	
1559						
1560	006312	020003		4\$: CMP	RO,R3	; AT THE 2ND INSTRUCTION
1561	006314	001020		BNE	6\$	; BR IF NO
1562	006316	020567	174404	CMP	R5,INSTR2	; DID IT CHANGE?
1563	006322	001412		BEQ	5\$	; BR IF NO
1564	006324	010567	174244	MOV	R5,\$BDDAT	
1565	006330	016767	174372	MOV	INSTR2,\$GDDAT	174234
1566	006336	010067	174226	MOV	RO,\$BDADR	
1567	006342	004767	000246	JSR	PC,\$HLT	
1568	006346	000002		2		
1569	006350	016710	174264	5\$: MOV	SVIN2,(RO)	; RESTORE THIS INSTRUCTION
1570	006354	000436		BR	7\$	
1571	006356	026704	174234	6\$: CMP	ROW,R4	; ON THE SAME ROW AS TSTADR?
1572	006362	001016		BNE	10\$	
1573	006364	026705	174234	CMP	TSTPAT,R5	; DID THE PATTERN CHANGE?

```

1574 006370 001430          BEQ      7$
1575 006372 010567 174176    MOV      R5,$BDDAT
1576 006376 016767 174222 174166  MOV      TSTPAT,$GDDAT
1577 006404 010067 174160    MOV      R0,$B0ADR
1578 006410 004767 000200    JSR      PC,$HLT
1579 006414 000001          I
1580 006416 000415          BR
1581 006420 020567 174202 10$:  CMP      R5,BCKGRD          ;DID BACK GROUND CHANGE
1582 006424 001412          BEQ      7$                ;BR IF NO
1583 006426 010567 174142    MOV      R5,$BDDAT
1584 006432 016767 174170 174132  MOV      BCKGRD,$GDDAT
1585 006440 010067 174124    MOV      R0,$B0ADR
1586 006444 004767 000144    JSR      PC,$HLT
1587 006450 000003          3
1588
1589 006452 062704 000020 7$:  ADD      #20,R4          ;GO TO THE NEXT ROW
1590 006456 042704 177017    BIC      #1C760,R4        ;DON'T LET IT GET TO BIG
1591 006462 001241          BNE
1592
1593 006464 062767 001000 177406 11$:  ADD      #1000,TCOL        ;GO TO THE NEXT COL
1594 006472 042767 140775 177400    BIC      #1C37002,TCOL    ;DON'T LET IT GET TO BIG
1595 006500 032767 020000 177372    BIT      #BIT13,TCOL
1596 006506 001627          BEQ      8$                ;IF MORE TO DO GO DO IT
1597 006510 010146          MOV      R1,-(SP)
1598 006512 012701 020002    MOV      #BIT13:BIT01,R1
1599 006516 074167 177356    XOR      R1,TCOL
1600 006522 012601          MOV      (SP)+,R1
1601 006524 000620          BR      8$
1602
1603 006526 012605          CHK1:  MOV      (SP)+,R5        ;POP STACK INTO R5
1604 006530 012604          MOV      (SP)+,R4        ;POP STACK INTO R4
1605 006532 012603          MOV      (SP)+,R3        ;POP STACK INTO R3
1606 006534 012602          MOV      (SP)+,R2        ;POP STACK INTO R2
1607 006536 105767 174012    TSTB    $ERFLG          ;WAS THERE ANY ERRORS?
1608 006542 001002          BNE      EXIT            ;IF YES EXIT
1609 006544 062716 000002    ADD      #2,(SP)         ;SET FOR NO ERRORS EXIT
1610 006550 000207          EXIT:  RTS      PC
1611
1612
1613
1614
1615 006552 010046          FORMAD: MOV      R0,-(SP)        ;SAVE R0
1616 006554 042700 160015    BIC      #1C17762,R0     ;STRIP EVERYTHING BUT THE ROW AND COL
1617 006560 062700 000020    ADD      #20,R0          ;ADD A ROW
1618 006564 032700 020000    BIT      #BIT13,R0       ;C4
1619 006570 001405          BEQ      1$              ;BR IF NO
1620 006572 010146          MOV      R1,-(SP)
1621 006574 012701 020002    MOV      #BIT13:BIT01,R1
1622 006600 074100          XOR      R1,R0
1623 006602 012601          MOV      (SP)+,R1
1624 006604 042716 017762 1$:  BIC      #17762,(SP)     ;STRIP THE ROW AND COL
1625 006610 052600          BIS      (SP)+,R0        ;FORM ADDRESS
1626 006612 000207          RTS      PC
1627
1628
1629
;*****

```

```

1630 :THIS ROUTINE IS THE HLT HANDLER
1631 :SW10=1 BELL ON ERROR
1632 :SW13=1 INHIBIT ERROR TYPEOUTS
1633 :SW15=1 HALT ON ERROR
1634 :GO TO TYPERR ON ERROR
1635 $HLT:
1636 006614 032737 002000 177570 BIT #SW10,2#SWR ;BELL ON ERROR?
1637 006622 001403 BEQ 1$ ;NO - SKIP
1638 006624 004767 000410 JSR PC,TYPE
1639 006630 002644 $BELL
1640 006632 005267 173714 1$: INC $ERTTL ;COUNT THE NUMBER OF ERRORS
1641 006636 001775 BEQ 1$ ;INSURE $ERTTL NOT=0
1642 006640 032737 020000 177570 BIT #SW13,2#SWR ;SKIP TYPEOUT IF SET
1643 006646 001015 BNE 2$ ;SKIP TYPEOUTS
1644 006650 011667 173710 MOV (SP), $HLTAD ;GET ADDRESS OF HLT INSTRUCTION
1645 006654 011767 173704 1,700 MOVB 2$HLTAD,$ITEMB ;SAVE THE ERROR ITEM CODE
1646 006662 162767 000002 173674 SUB #2,$HLTAD
1647 006670 004767 000046 JSR PC,TYPERR ;GO TO USER ERROR ROUTINE
1648 006674 004767 000340 JSR PC,TYPE
1649 006700 002647 $CRLF
1650 006702 005737 177570 2$: TST 2#SWR ;HALT ON ERROR
1651 006706 100001 BPL 3$ ;SKIP IF CONTINUE
1652 006710 000000 HALT ;HALT ON ERROR!
1653 006712 112767 177777 173634 3$: MOVB #-1,$ERFLG
1654 006720 062716 000002 ADD #2,(SP)
1655 006724 032737 001000 177570 BIT #BIT09,2#SWR
1656 006732 001002 BNE 4$
1657 006734 016716 173702 MOV ESCAPE,(SP)
1658 006740 000207 4$: RTS PC
1659
1660 006742 010046 TYPERR: MOV RO,-(SP) ;PUSH RO ON STACK
1661 006744 010146 MOV R1,-(SP) ;PUSH R1 ON STACK
1662 006746 116700 173610 MOVB $ITEMB,RO ;PICKUP THE ITEM NUMBER
1663 006752 005300 DEC RO ;FORM AN INDEX FOR THE TABLE
1664 006754 006300 ASL RO
1665 006756 006300 ASL RO
1666 006760 006300 ASL RO
1667 006762 010746 MOV PC,-(SP)
1668 006764 062716 173710 ADD #$ERRTB-.,(SP)
1669 006770 062600 ADD (SP)+,RO
1670 006772 012067 000012 MOV (RO)+,1$ ;PICKUP THE ERROR MESSAGE
1671 006776 004767 000236 JSR PC,TYPE
1672 007002 002647 $CRLF
1673 007004 004767 000230 JSR PC,TYPE
1674
1675 007010 000000 1$: 0 ;ERROR MESSAGE POINTER GOES HERE
1676 007012 012067 000014 MOV (RO)+,2$ ;PICKUP THE DATA HEADER POINTER
1677 007016 032737 000400 177570 BIT #BIT08,2#SWR ;TYPE THE DATA HEADER?
1678 007024 001003 BNE 11$ ;BR IF NO
1679 007026 004767 000206 JSR PC,TYPE
1680 007032 000000 2$: 0
1681 007034 012001 11$: MOV (RO)+,R1 ;GET THE DATA STRING POINTER
1682 007036 011000 MOV (RO),RO ;GET THE DATA FORMAT POINTER
1683 007040 010746 MOV PC,-(SP)
1684 007042 062716 170736 ADD #-1,(SP)
1685 007046 061600 ADD (SP),RO

```

```

1686 007050 062601          ADD      (SP)+, R1
1687 007052 016767 173552 173516  MOV      INSTAD, $TMP0      ;SETUP THE COLUMN FOR TYPING
1688 007060 000367 173512          SWAB     $TMP0
1689 007064 006267 173506          ASR      $TMP0
1690 007070 042767 177740 173500  BIC      #1C37, $TMP0
1691 007076 016767 173526 173474  MOV      INSTAD, $TMP1      ;SET THE ROW FOR TYPING
1692 007104 006267 173470          ASR      $TMP1
1693 007110 006267 173464          ASR      $TMP1
1694 007114 006267 173460          ASR      $TMP1
1695 007120 006267 173454          ASR      $TMP1
1696 007124 042767 177740 173446  BIC      #1C37, $TMP1
1697 007132 016767 173472 173442  MOV      INSTAD, $TMP2      ;SETUP PRECHARGE FOR TYPING
1698 007140 006267 173436          ASR      $TMP2
1699 007144 006267 173432          ASR      $TMP2
1700 007150 042767 177774 173424  BIC      #1C3, $TMP2
1701 007156 000410          BR       5$
1702 007160 005711          10$:    TST      (R1)          ;TERMINATOR?
1703 007162 001003          BNE     3$          ;BR IF NO
1704 007164 012601          MOV      (SP)+, R1        ;POP STACK INTO R1
1705 007166 012600          MOV      (SP)+, R0        ;POP STACK INTO R0
1706 007170 000207          RTS     PC
1707 007172 004067 000074          3$:    JSR      R0, $SPACE      ;GO TYPE SOME SPACES
1708 007176 000000          4$:    .WORD  0          ;HOW MANY GOES HERE
1709 007200 112067 000030          5$:    MOVB   (R0)+, 6$        ;PICKUP NUMBER OF DIGITS TO TYPE
1710 007204 112067 000025          MOVB   (R0)+, 7$        ;PICKUP THE LEADING ZEROS FLAG
1711 007210 112067 177762          MOVB   (R0)+, 4$        ;PICKUP THE NUMBER OF SPACES
1712 007214 012146          MOV      (R1)+, -(SP)    ;PICKUP THE DATA
1713 007216 010746          MOV      PC, -(SP)
1714 007220 062716 170560          ADD     #-, (SP)
1715 007224 062616          ADD     (SP)+, (SP)
1716 007226 013646          MOV     2(SP)+, -(SP)
1717 007230 004067 000064          JSR     R0, $B2OCT      ;GO TYPE AN OCTAL DIGIT
1718 007234 000          6$:    .BYTE  0          ;HOW MANY DIGITS
1719 007235 000          7$:    .BYTE  0          ;LEADING ZERO SWITCH
1720 007236 000750          BR      10$          ;LOOP
1721
1722          ;*****
1723
1724          ;THIS IS A SMALL ROUTINE USED TO MAKE THE TYPE POINTERS POSITION INDEPENT
1725
1726 007240 010746          TYPE:   MOV      PC, -(SP)      ;GET THE POINTER
1727 007242 062716 170536          ADD     #-, (SP)          ;GET THE PC
1728 007246 067616 000002          ADD     2(SP), (SP)      ;STRIP THE RELATIVE ADDRESS
1729 007252 012667 000004          MOV     (SP)+, 1$        ;AND SAVE IT FOR TYPE OUT
1730 007256 004767 171626          JSR     PC, $TYPE        ;GO TYPE TO
1731 007262 000000          1$:    .WORD  0          ;POINTER TO MESSAGE
1732 007264 062716 000002          ADD     #2, (SP)          ;SETUP TO RETURN
1733 007270 000207          RTS     PC          ;GO BACK
1734
1735          ;*****
1736          ;ROUTINE TO TYPE SPACES ON THE TTY
1737          ;CALL:
1738          ;       JSR      R0, $SPACE
1739          ;       NUMBER OF SPACES
1740          ;       RETURN HERE
1741

```

```

1742 007272 012067 000002 $SPACE: MOV (R0)+,2$ ;GET NUMBER OF SPACES DESIRED
1743 ;AND SET RETURN ADR.
1744 007276 005327 1$: DEC (PC)+ ;FINISHED?
1745 007300 000000 2$: 0
1746 007302 002001 BGE 3$ ;BR IF NO
1747 007304 000200 RTS R0 ;GO HOME
1748 007306 3$:
1749 007306 004767 177726 JSR PC,TYPE
1750 007312 007316 4$:
1751 007314 000770 BR 1$
1752 007316 040 000 4$: .BYTE 40,0 ;STORAGE FOR ONE SPACE AND TERMINATOR
1753 ;*****
1754 ;BINARY TO OCTAL (ASCII) AND TYPE
1755 ;$B2OCT---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
1756 ;CALL:
1757 ; MOV NUM,-(SP) ;NUMBER TO BE TYPED
1758 ; JSR R0,$B2OCT ;CALL FOR TYPEOUT
1759 ; .BYTE N ;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
1760 ; .BYTE M ;M=0 TO 3
1761 ; 0=SUPPRESS LEADING ZEROS
1762 ; 1=TYPE LEADING ZEROS
1763 ; 2=SUPPRESS LEADING ZEROS
1764 ; WITH PHYSICAL ADDRESS
1765 ; 3=TYPE LEADING ZEROS
1766 ; WITH PHYSICAL ADDRESS
1767
1768 ;$B201----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST $B5OCT OR $B2016
1769 ;CALL:
1770 ; MOV NUM,-(SP)
1771 ; JSR R0,$B201
1772
1773 ;$B2016---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
1774 ;CALL:
1775 ; MOV NUM,-(SP)
1776 ; JSR R0,$B2016
1777
1778 007320 112067 000267 $B2OCT: MOV (R0)+,$SOMODE+1 ;PICKUP THE NUMBER OF DIGITS TO TYPE
1779 007324 112067 000261 MOV (R0)+,$OFILL ;GET THE ZERO FILL SWITCH
1780 007330 000406 BR $B201
1781 007332 112767 000001 000251 $B2016: MOV #1,$OFILL ;SET THE ZERO FILL SWITCH
1782 007340 112767 000006 000245 MOV #6,$SOMODE+1 ;SET FOR SIX(6) DIGITS
1783 007346 112767 000005 000234 $B201: MOV #5,$OCNT ;SET THE ITERATION COUNT
1784 007354 010346 MOV R3,-(SP) ;SAVE R3
1785 007356 010446 MOV R4,-(SP) ;SAVE R4
1786 007360 010546 MOV R5,-(SP) ;SAVE R5
1787 007362 116704 000225 MOV $SOMODE+1,R4 ;GET THE NUMBER OF DIGITS TO TYPE
1788 007366 005404 NEG R4
1789 007370 062704 000006 ADD #6,R4 ;SUBTRACT IT FOR MAX. ALLOWED
1790 007374 110467 000212 MOV R4,$SOMODE ;SAVE IT FOR USE
1791 007400 116704 000205 MOV $OFILL,R4 ;GET THE ZERO FILL SWITCH
1792 007404 042704 177776 BIC #177776,R4
1793 007410 016605 000010 MOV 10(SP),R5 ;PICKUP THE INPUT NUMBER
1794 007414 005003 CLR R3 ;CLEAR THE OUTPUT WORD
1795 007416 132767 000002 000165 BITB #2,$OFILL
1796 007424 001424 BEQ 1$
1797 007426 005767 171670 TST $K111

```

1798	007432	100021		BPL	1\$	
1799	007434	032777	000001 170440	BIT	#1,RSRO	
1800	007442	001415		BEQ	1\$	
1801	007444	017746	173216	MOV	2PAGE,-(SP)	
1802	007450	042705	160000	BIC	#160000,R5	
1803	007454	006316		ASL	(SP)	
1804	007456	006316		ASL	(SP)	
1805	007460	006316		ASL	(SP)	
1806	007462	006316		ASL	(SP)	
1807	007464	006316		ASL	(SP)	
1808	007466	006103		ROL	R3	
1809	007470	006316		ASL	(SP)	
1810	007472	006103		ROL	R3	
1811	007474	062605		ADD	(SP)+,R5	
1812	007476	006105		1\$: ROL	R5	: ROTATE MSB INTO "C"
1813	007500	000404		BR	3\$	: GO DO MSC
1814	007502	006105		2\$: ROL	R5	: FORM THIS DIGIT
1815	007504	006105		ROL	R5	
1816	007506	006105		ROL	R5	
1817	007510	010503		MOV	R5,R3	
1818	007512	006103		3\$: ROL	R3	: GET LSB OF THIS DIGIT
1819	007514	105367	000072	DECB	\$OMODE	: TYPE THIS DIGIT?
1820	007520	100017		BPL	7\$	: BR IF NO
1821	007522	042703	177770	BIC	#177770,R3	: GET RID OF JUNK
1822	007526	001002		BNE	4\$	: TEST FOR 0
1823	007530	005704		TST	R4	: SUPPRESS THIS 0?
1824	007532	001403		BEQ	5\$	: BR IF YES
1825	007534	005204		4\$: INC	R4	: DON'T SUPPRESS ANYMORE 0'S
1826	007536	052703	000060	BIS	#'0,R3	: MAKE THIS DIGIT ASCII
1827	007542	052703	000040	5\$: BIS	#' ,R3	: MAKE ASCII IF NOT ALREADY
1828	007546	110367	000034	MOVB	R3,8\$	: SAVE FOR TYPING
1829	007552	004767	177462	JSR	PC,TYPE	
1830	007556	007606		8\$		
1831	007560	105367	000024	7\$: DECB	\$OCNT	: COUNT BY 1
1832	007564	003346		BGT	2\$	: BR IF MORE TO DO
1833	007566	002402		BLT	6\$	: BR IF DONE
1834	007570	005204		INC	R4	: INSURE LAST DIGIT ISN'T A BLANK
1835	007572	000743		BR	2\$	: GO DO THE LAST DIGIT
1836	007574	012605		6\$: MOV	(SP)+,R5	: RESTORE R5
1837	007576	012604		MOV	(SP)+,R4	: RESTORE R4
1838	007600	012603		MOV	(SP)+,R3	: RESTORE R3
1839	007602	012616		MOV	(SP)+,(SP)	: SET THE STACK FOR RETURNING
1840	007604	000200		RTS	RO	: RETURN
1841	007606	000		8\$: .BYTE	0	: STORAGE FOR ASCII DIGIT
1842	007607	000		.BYTE	0	: TERMINATOR FOR TYPE ROUTINE
1843	007610	000		\$OCNT: .BYTE	0	: OCTAL DIGIT COUNTER
1844	007611	000		\$OFILL: .BYTE	0	: ZERO FILL SWITCH
1845	007612	000000		\$OMODE: 0		: NUMBER OF DIGITS TO TYPE

```

1848
1849
1850
1851 007614 042524 052123 050040
1852 007622 052101 042524 047122
1853 007630 005015 000
1854 007633 111 051516 051124
1855 007640 041525 044524 047117
1856 007646 041440 040510 043516
1857 007654 042105 005015 000
1858 007661 102 041501 020113
1859 007666 051107 052517 042116
1860 007674 041440 040510 043516
1861 007702 042105 005015 000
1862 007707 105 051122 051117
1863 007714 020040 043040 044501
1864 007722 044514 043516 005015
1865 007730 041520 020040 020040
1866 007736 020040 042101 051104
1867 007744 051505 020123 054105
1868 007752 042520 052103 020040
1869 007760 041522 023526 020104
1870 007766 020040 047503 020114
1871 007774 051040 053517 020040
1872 010002 051120 041505 040510
1873 010010 043522 006505 000012
1874 010016 051105 047522 020122
1875 010024 020040 047111 052123
1876 010032 020040 020040 054105
1877 010040 042520 052103 020040
1878 010046 041522 023526 006504
1879 010054 012
1880 010055 120 020103 020040
1881 010062 020040 040440 042104
1882 010070 042522 051523 042040
1883 010076 052101 020101 020040
1884 010104 042040 052101 020101
1885 010112 020040 041440 046117
1886 010120 020040 047522 020127
1887 010126 050040 042522 044103
1888 010134 051101 042507 005015
1889 010142 000
1890 010143 105 051122 051117
1891 010150 020040 041040 045503
1892 010156 051107 020104 042440
1893 010164 050130 041505 020124
1894 010172 051040 053103 042047
1895 010200 005015
1896 010202 041520 020040 020040
1897 010210 020040 042101 051104
1898 010216 051505 020123 040504
1899 010224 040524 020040 020040
1900 010232 040504 040524 020040
1901 010240 020040 047503 020114

```

```

:*****
:*****
:ASCII MESSAGES
EM1: .ASCIZ /TEST PATTERN/(15)(12)
EM2: .ASCIZ /INSTRUCTION CHANGED/(15)(12)
EM3: .ASCIZ /BACK GROUND CHANGED/(15)(12)
DM1: .ASCII /ERROR FAILING/(15)(12)
      .ASCIZ /PC ADDRESS EXPECT RCV'D COL ROW PRECHARGE/(15)(12)
DM2: .ASCII /ERROR INST EXPECT RCV'D/(15)(12)
      .ASCIZ /PC ADDRESS DATA DATA COL ROW PRECHARGE/(15)(12)
DM3: .ASCII /ERROR BCKGRD EXPECT RCV'D/(15)(12)
      .ASCIZ /PC ADDRESS DATA DATA COL ROW PRECHARGE/(15)(12)

```



1902	010246	051040	053517	020040
1903	010254	051120	041505	040510
1904	010262	043522	006505	000012
1905				
1906	010270	002564	002570	002572
1907	010276	002574	002576	002600
1908	010304	002602	000000	
1909	010310	006	001	002
1910	010313	006	003	002
1911	010316	006	001	002
1912	010321	006	001	003
1913	010324	002	000	003
1914	010327	002	000	005
1915	010332	002	000	000
1916	010335	105	042116	050040
1917	010342	051501	006523	000012
1918	010350	042523	020124	053523
1919	010356	052111	044103	051505
1920	010364	043040	051117	047440
1921	010372	042520	040522	044524
1922	010400	043516	046440	042117
1923	010406	020105	042522	042506
1924	010414	020122	042523	052103
1925	010422	047511	020116	027064
1926	010430	006461	012	
1927	010433	040	042511	020056
1928	010440	044520	045503	050125
1929	010446	020123	051117	042040
1930	010454	047522	051520	047440
1931	010462	020122	047502	044124
1932	010470	052040	042510	020116
1933	010476	044510	020124	047503
1934	010504	052116	047111	042525
1935	010512	005015	000	
1936	010515	123	052105	051440
1937	010522	044527	041524	042510
1938	010530	006523	000012	
1939	010534	054524	042520	021040
1940	010542	052123	051101	044524
1941	010550	043516	041040	047101
1942	010556	021113	047040	046525
1943	010564	042502	020122	047101
1944	010572	020104	051103	006440
1945	010600	012		
1946	010601	122	043105	051105
1947	010606	052040	020117	042523
1948	010614	052103	047511	020116
1949	010622	027064	027063	020061
1950	010630	042502	047506	042522
1951	010636	041440	047510	051517
1952	010644	047111	020107	005015
1953	010652	000		
1954	010653	123	040524	052122
1955	010660	047111	020107	040502
1956	010666	045516	006477	000012
1957	010674	054524	042520	047040

.EVEN  
DT1: .WORD \$HLTAD,\$BDOADR,\$GDDAT,\$BDDAT,\$STMP0,\$STMP1,\$STMP2,0

DF1: .BYTE 6,1,2  
.BYTE 6,3,2  
.BYTE 6,1,2  
.BYTE 6,1,3  
.BYTE 2,0,3  
.BYTE 2,0,5  
.BYTE 2,0,0

EPASS: .ASCIZ /END PASS/<15><12>

OPMODE: .ASCII /SET SWITCHES FOR OPERATING MODE REFER SECTION 4.1/<15><12>

.ASCIZ / IE. PICKUPS OR DROPS OR BOTH THEN HIT CONTINUE/<15><12>

OPMOD2: .ASCIZ /SET SWITCHES/<15><12>

STARTB: .ASCII /TYPE "STARTING BANK" NUMBER AND CR /<15><12>

.ASCIZ /REFER TO SECTION 4.3.1 BEFORE CHOOSING /<15><12>

START2: .ASCIZ /STARTING BANK?/<15><12>

NUMBER: .ASCIZ /TYPE NUMBER OF 4K BANKS TO BE TESTED AND CR /<15><12>

1958	010702	046525	042502	020122	
1959	010710	043117	032040	020113	
1960	010716	040502	045516	020123	
1961	010724	047524	041040	020105	
1962	010732	042524	052123	042105	
1963	010740	040440	042116	041440	
1964	010746	020122	005015	000	
1965	010753	043	047440	020106	NUMB2: .ASCIZ /# OF BANKS?(<15><12>)
1966	010760	040502	045516	037523	
1967	010766	005015	000		
1968	010771	114	051501	020124	MEMEND: .ASCIZ /LAST MEMORY ADDRESS IS /
1969	010776	042515	047515	054522	
1970	011004	040440	042104	042522	
1971	011012	051523	044440	020123	
1972	011020	000			
1973	011021	120	051101	052111	PAERR: .ASCIZ /PARITY ERROR DETECTED AT LOC. /
1974	011026	020131	051105	047522	
1975	011034	020122	042504	042524	
1976	011042	052103	042105	040440	
1977	011050	020124	047514	027103	
1978	011056	020040	000		
1979	011061	111	020116	040520	NOPARE: .ASCIZ /IN PARITY TEST NO PARITY ERPOR DECTED/<15><12>
1980	011066	044522	054524	052040	
1981	011074	051505	020124	047516	
1982	011102	050040	051101	052111	
1983	011110	020131	051105	047522	
1984	011116	020122	042504	052103	
1985	011124	042105	005015	000	
1986	011131	120	051101	052111	PARERR: .ASCIZ /PARITY ERROR HAS OCCURED SO PARITY TEST HAS STARTED/<15><12>
1987	011136	020131	051105	047522	
1988	011144	020122	040510	020123	
1989	011152	041517	052503	042522	
1990	011160	020104	047523	050040	
1991	011166	051101	052111	020131	
1992	011174	042524	052123	044040	
1993	011202	051501	051440	040524	
1994	011210	052122	042105	005015	
1995	011216	000			
1996	011217	123	051117	054522	SORRY: .ASCIZ /SORRY LAST INPUT INVALID TRY AGAIN/<15><12>
1997	011224	046040	051501	020124	
1998	011232	047111	052520	020124	
1999	011240	047111	040526	044514	
2000	011246	020104	051124	020131	
2001	011254	043501	044501	006516	
2002	011262	000012			
2003	011264	051120	043517	040522	PROTO: .ASCII /PROGRAM RELOCATED/<15><12>
2004	011272	020115	042522	047514	
2005	011300	040503	042524	006504	
2006	011306	012			
2007	011307	122	051505	040524	.ASCIZ /RESTART ADDRESS /
2008	011314	052122	040440	042104	
2009	011322	042522	051523	020040	
2010	011330	000040			
2011	011332	047524	050040	052125	PROBAK: .ASCIZ /TO PUT PROGRAM BACK TO FIRST 4K START /
2012	011340	050040	047522	051107	
2013	011346	046501	041040	041501	

2014	011354	020113	047524	043040	
2015	011362	051111	052123	032040	
2016	011370	020113	052123	051101	
2017	011376	020124	000040		
2018	011402	047524	051040	051505	RELOD: .ASCIZ /TO RESTORE LOADER START /
2019	011410	047524	042522	046040	
2020	011416	040517	042504	020122	
2021	011424	052123	051101	020124	
2022	011432	000040			
2023	011434	050040	052514	020123	RELOD2: .ASCIZ / PLUS RELOCATION FACTOR/<15><12>
2024	011442	042522	047514	040503	
2025	011450	044524	047117	043040	
2026	011458	041501	047524	006522	
2027	011464	000012			
2028	011466	047514	042101	051105	FIN: .ASCIZ /LOADER RESTORED/<15><12>
2029	011474	051040	051505	047524	
2030	011502	042522	006504	000012	
2031					.EVEN
2032	011510	000000			LAST: .WORD 0
2033		000001			.END





PAGE	002666	620*	853*	923*	1039*	1352*	1801										
PARCSR=	172100	719*	727														
PARERR	011131	739	1986*														
PARSRV	002254	723	738*	782													
PARVEC=	000114	406*	723*	724*	742*	783*											
PASFLG	002652	847*	913*	1031*	1140*	1161*	1168*	1367*	1369*	1444*	1447						
PC	=%000007	338*	479	485*	526*	527*	534*	539*	563	566	592	627	636	659*			
		682*	693*	698*	701*	706*	708*	734*	738*	740	743	746	768*	779*			
		781	784	788	794*	895*	899*	911	915*	918*	919*	932*	934*	938*			
		940*	943*	977	982*	989*	992	1004*	1011*	1013*	1020*	1029	1034*	1035*			
		1048*	1087	1092*	1098*	1104*	1106*	1112*	1217	1229	1235	1291*	1299	1304*			
		1316*	1452*	1477*	1505*	1524	1544*	1555*	1567*	1578*	1586*	1610*	1626*	1638*			
		1647*	1648*	1658*	1667	1671*	1673*	1679*	1683	1706*	1713	1726	1730*	1733*			
		1744*	1749*	1829*													
PCHRG	002620	833*	1150*	1199	1321*	1322*											
PROJBAK	011332	1014	2011*														
PROTO	011264	1005	2003*														
PS	= 177776	323*	324														
PSW	= 177776	324*															
PWRVEC=	000024	403*															
REDONE	002656	849*	1067*	1415*	1416	1423*	1436*										
RELOD	011402	983	1093	2018*													
RELOD2	011434	989	1099	2023*													
RESTOR	000300	465*	1016														
RESVEC=	000010	398*															
ROW	002616	832*	1149*	1201	1467	1571											
PO	=%000000	329*	519	520*	521	524*	560	634*	639*	641*	646*	650	654*	727*			
		730*	749*	764	771	772*	801*	893*	897*	926*	926*	981*	985*	999*			
		1002*	1008*	1017*	1042*	1089*	1091*	1095*	1118*	1119*	1125*	1203*	1204*	1205*			
		1206*	1207	1208	1217*	1219*	1239*	1241*	1243*	1255*	1260	1267*	1268	1281			
		1295	1308*	1310*	1354*	1355	1356	1357	1358	1359	1360	1361	1362	1405*			
		1414*	1465*	1466*	1467	1476*	1478	1490	1491	1492	1493*	1494*	1495*	1497			
		1498*	1499	1517*	1520	1522	1529*	1530*	1531*	1532	1535	1536	1543	1548			
		1554	1557*	1560	1566	1569*	1577	1585	1615	1616*	1617*	1618	1622*	1625*			
		1660	1662*	1663*	1664*	1665*	1666*	1669*	1670	1676	1681	1682*	1685*	1705*			
		1707*	1709	1710	1711	1717*	1742	1747*	1778	1779	1840*						
R1	=%000001	330*	465*	468	482*	484*	559	562*	640*	642	643	645*	649	655*			
		894*	898*	1000*	1001	1121*	1132*	1229*	1231*	1235*	1237*	1239	1241	1243			
		1257*	1260*	1268*	1269*	1279	1297*	1309*	1406*	1413	1437*	1439*	1442*	1515*			
		1536	1597	1598*	1599	1600*	1620	1621*	1622	1623*	1661	1681*	1686*	1702			
		1704*	1712														
R2	=%000002	331*	466*	468*	481*	483*	728*	731*	752*	754	757	759	970*	1003*			
		1022	1083*	1122*	1123*	1127	1128	1130*	1133*	1134	1259*	1261*	1270*	1273*			
		1274*	1275	1306*	1307	1308	1407*	1413*	1438*	1440*	1443*	1463	1479*	1480			
		1483	1486	1499	1504*	1511	1516*	1517	1518	1532	1548	1606*					
R3	=%000003	332*	467*	469*	558	571*	572	573	574	575	576	577	578	579			
		592*	594*	595*	596*	597*	598*	599	610*	611	612	613	614	656*			
		974*	975*	976	980	1084*	1085*	1086	1090	1120*	1124*	1126	1129*	1131*			
		1135	1142	1271*	1278	1441*	1443	1464	1478*	1480	1482*	1483	1485*	1486			
		1488	1503*	1512	1518*	1519*	1560	1605*	1784	1794*	1808*	1810*	1817*	1818*			
		1821*	1826*	1827*	1828	1838*											
R4	=%000004	333*	479*	480*	483	557	603*	608*	620	621*	631*	632	634	657*			
		958*	959*	960	962*	964*	965*	966*	967*	968*	969*	970	974	995			
		997*	998	1001*	1071*	1072*	1073	1075*	1077*	1078*	1079*	1080*	1081*	1082*			
		1083	1084	1272*	1280	1513	1520*	1521*	1530	1571	1589*	1590*	1604*	1785			
		1787*	1788*	1789*	1790	1791*	1792*	1823	1825*	1834*	1837*						









L04

TEST MOS MEMORY MACY:1 27(732) 22-SEP-76 14:48 PAGE 52  
DCMS88.P11 CROSS REFERENCE TABLE -- MACRO NAMES

HLT 321#  
SCOPE 322#

ADD	480	525	567	628	631	637	640	641	697	741	744	747	782	785	789
	800	912	975	978	996	997	1003	1007	1016	1030	1052	1085	1088	1123	1124
	1130	1131	1219	1231	1237	1273	1321	1326	1346	1439	1440	1471	1482	1485	1501
	1519	1525	1589	1593	1609	1617	1654	1668	1669	1684	1685	1686	1714	1715	1727
	1728	1732	1789	1811											
ASL	712	713	714	731	1664	1665	1666	1803	1804	1805	1806	1807	1809		
ASR	686	687	1689	1692	1693	1694	1695	1698	1699						
BCC	732														
BCS	1327														
BEQ	516	600	705	1111	1170	1197	1227	1277	1301	1396	1417	1468	1481	1484	1487
	1500	1533	1540	1551	1563	1574	1582	1596	1619	1637	1641	1796	1800	1824	
BGE	1746														
BGT	633	1832													
BHI	961	1074	1143	1338	1489										
BIC	598	678	703	711	993	1193	1195	1200	1202	1223	1269	1274	1322	1466	1521
	1523	1529	1590	1594	1616	1624	1690	1696	1700	1792	1802	1821			
BIS	570	715	1198	1204	1205	1206	1279	1530	1531	1625	1826	1827			
BIT	1110	1159	1169	1196	1276	1319	1324	1365	1450	1595	1618	1636	1642	1655	1677
	1799														
BITB	1795														
BLC	1137	1330	1351												
BLOS	1340														
BLT	533	1833													
BNE	522	529	644	680	692	972	991	994	1069	1160	1163	1282	1320	1323	1325
	1366	1448	1451	1537	1549	1561	1572	1591	1608	1643	1656	1678	1703	1822	
BPL	537	565	652	676	751	776	922	930	1038	1046	1400	1455	1651	1798	1820
BR	518	535	604	635	688	710	716	765	796	1023	1138	1167	1174	1178	1233
	1293	1305	1317	1318	1331	1345	1368	1418	1473	1527	1546	1558	1570	1580	1601
	1701	1720	1751	1780	1813	1835									
CLC	593	963	1076												
CLR	582	621	639	653	674	733	749	753	755	756	777	893	897	913	931
	945	1031	1047	1067	1143	1149	1150	1164	1165	1168	1171	1172	1177	1181	1182
	1353	1363	1369	1401	1410	1411	1456	1794							
CLRB	1245														
CMP	599	632	643	704	960	1073	1136	1142	1281	1329	1337	1339	1350	1467	1480
	1483	1486	1488	1499	1532	1536	1539	1548	1550	1560	1562	1571	1573	1581	
CMPB	528	679	691												
COM	1140	1161	1367	1415	1444										
DEC	1060	1133	1663	1744											
DECB	532	1819	1831												
EMT	321														
HALT	487	517	1109	1652											
INC	1119	1449	1640	1825	1834										
IOT	322														
JMP	449	451	452	470	790	1022	1141	1332	1370	1432	1433	1445	1457		
JSR	485	527	534	682	693	701	706	708	738	768	772	778	794	915	918
	919	926	932	934	938	940	943	982	985	988	1004	1008	1011	1013	1017
	1020	1034	1035	1042	1048	1092	1095	1098	1104	1106	1112	1291	1304	1316	1452
	1477	1544	1555	1567	1578	1586	1638	1647	1648	1671	1673	1679	1707	1717	1730
	1749	1829													
MOV	465	466	467	468	479	481	482	483	519	520	524	530	556	557	558
	559	560	561	562	566	568	571	572	573	574	575	576	577	578	579
	583	584	585	586	587	588	589	592	603	608	609	610	611	612	613
	614	615	616	620	626	627	629	634	636	638	647	648	649	650	654
	655	656	657	658	681	696	723	724	725	726	727	728	730	740	742
	743	745	746	748	752	754	757	758	759	760	770	771	781	783	784



TEST MOS MEMORY MAC11 (27.732) 22-SEP-75 14:48 PAGE 56  
DCMSBB.P11 CROSS REFERENCE TABLE -- PERMANENT SYMBOLS

ERRORS DETECTED: 0  
DEFAULT GLOBALS GENERATED: 0

\* DCMSBB.SEG/SOL/CRF/PAGNUM/NL:TOC=DCMSBB  
RUN-TIME: 5 12 3 SECONDS  
RUN-TIME RATIO: 87/20=4.2  
CORE USED: BK (15 PAGES)