IDENTIFICATION

PRODUCT CODE	MAINDEC-11-DCMSB-B-D
PRODUCT NAME	MOS SEMICONDUCTOR MEMORY DIAGNOSTIC
DATE CREATED	APRIL 23, 1973
MAINTAINER	DIAGNOSTIC GROUP
AUTHOR	S. MALLICK

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
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 1 = BELL ON ERROR
9	0 = PRINT FIRST ERROR PER PRECHARGE 1 = PRINT ALL ERRORS PER PRECHARGE
8	INHIBIT PRINTING DATA HEADER
7	0=TEST READ AND WRITE FROM OUTSIDE OF THE CHIP 1=TEST INSTRUCTIONS IN THE CHIP
1	LOCK ON PRESENT MOD
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
SECOND 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
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)		
1	020000 (4K)	20	400000 (64K)
2	040000 (8K)	21	420000 (68K)
3	060000 (12K)	22	440000 (72K)
4	100000 (16K)	23	460000 (76K)
5	120000 (20K)	24	500000 (80K)
6	140000 (24K)	25	520000 (84K)
7	160000 (28K)	26	540000 (88K)
10	200000 (32K)	27	560000 (92)
11	220000 (36K)	30	600000 (96K)
12	240000 (40K)	31	620000 (100K)
13	260000 (44K)	32	640000 (104K)
14	300000 (48K)	33	660000 (108K)
15	320000 (52K)	34	700000 (112K)
16	340000 (56K)	35	720000 (116K)
17	360000 (60K)	36	740000 (120K)

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 APPROPIATE 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 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 TSTTPAT , IN ANOTHER LOCATION CALLED TEST-LOCATION
IT WILL WRITE A CLR RØ (OR ALL "Ø" 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 1ØØØ 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,Ø OPERATING PROCEDURE

5,1 OPERATIONAL SWITCH SETTINGS

SEE 4,1

6,Ø 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 PC	BCKGRD ADDRESS	EXPECT DATA	RCV'D DATA	COL	ROW	PRECHARGE
1Ø3166	ØØ65ØØ	177777	177577	6	23	Ø

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
VECTER AREA THE PROGRAM MUST BE RESTARTED. IF THE
PROGRAM HALTS IN THE MAIN FLOW, CONSULT THE LISTING
IF NO MESSAGE IS TYPED OUT.

7,Ø RESTRICTIONS

RELOCATION CANNOT BE DONE INTO 16ØØØØ

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 MSCELLANEOUS

8.1 EXECUTION TIME

APPOXIMATELY 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
                        ;                       1 = BELL ON ERROR
                        ;          9            0 = PRINT FIRST ERROR PER PRECHARGE
                        ;                       1 = PRINT ALL ERRORS PER PRECHARGE
                        ;          8            INHIBIT PRINTING DATA HEADER
                        ;          7            0=TEST READ AND WRITE FROM OUTSIDE OF THE CHIP
                        ;                       1=TEST INSTRUCTIONS IN THE CHIP
                        ;          1            LOCK ON PRESENT MOD
                        ;          0            LOCK ON PRESENT PRECHARGE
                        ;*********************************************************************


                        ;BASIC DEFINITIONS
                        .EQUIV  EMT,MLT                 ;BASIC DEFINITION OF ERROR CALL
                        .EQUIV  IOT,SCOPE               ;BASIC DEFINITION OF SCOPE CALL
 177776                 PS=     177776                 ;PROCESSOR STATUS WORD
                        .EQUIV  PS,PSW
 177570                 SWR=    177570                 ;SWITCH REGISTER
 177570                 DISPLAY=SWR

                        ;REGISTER DEFINITION
 000000                 R0=     %0                     ;GENERAL REGISTER
 000001                 R1=     %1                     ;GENERAL REGISTER
 000002                 R2=     %2                     ;GENERAL REGISTER
 000003                 R3=     %3                     ;GENERAL REGISTER
 000004                 R4=     %4                     ;GENERAL REGISTER
 000005                 R5=     %5                     ;GENERAL REGISTER
 000006                 R6=     %6                     ;GENERAL REGISTER
 000007                 R7=     %7                     ;GENERAL REGISTER
                        .EQUIV  R6,SP                  ;STACK POINTER
                        .EQUIV  R7,PC                  ;PROGRAM COUNTER

                        ;SWITCH DEFINITION
 100000                 SW15=   100000
 040000                 SW14=   40000
```



```
 020000                 SW13=   20000
 010000                 SW12=   10000
 004000                 SW11=   4000
 002000                 SW10=   2000
 001000                 SW09=   1000
 000400                 SW08=   400
 000200                 SW07=   200
 000100                 SW06=   100
 000040                 SW05=   40
 000020                 SW04=   20
 000010                 SW03=   10
 000004                 SW02=   4
 000002                 SW01=   2
 000001                 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

                        ;MISCELLANEOUS BIT ASSIGNMENT
 100000                 BIT15=  100000
 040000                 BIT14=  40000
 020000                 BIT13=  20000
 010000                 BIT12=  10000
 004000                 BIT11=  4000
 002000                 BIT10=  2000
 001000                 BIT09=  1000
 000400                 BIT08=  400
 000200                 BIT07=  200
 000100                 BIT06=  100
 000040                 BIT05=  40
 000020                 BIT04=  20
 000010                 BIT03=  10
 000004                 BIT02=  4
 000002                 BIT01=  2
 000001                 BIT00=  1
                        .EQUIV  BIT09,BIT9
                        .EQUIV  BIT08,BIT8
                        .EQUIV  BIT07,BIT7
                        .EQUIV  BIT06,BIT6
                        .EQUIV  BIT05,BIT5
                        .EQUIV  BIT04,BIT4
                        .EQUIV  BIT03,BIT3
                        .EQUIV  BIT02,BIT2
                        .EQUIV  BIT01,BIT1
                        .EQUIV  BIT00,BIT0

                        ;VECTOR ADDRESSES
```

```
  109           000004                  ERRVEC= 4
  110           000010                  RESVEC= 10
  111           000014                  TBITVEC=14
  112           000014                  TRTVEC= 14
  113           000014                  BPTVEC= 14
  114           000020                  IOTVEC= 20
  115           000024                  PWRVEC= 24
  116           000030                  EMTVEC= 30
  117           000034                  TRAPVEC=34
  118           000114                  PARVEC=114
  119           000250                  MMVEC=250
```



```
  120
  121           000100                          .=100
  122  000100  000000                  FIRST:  .WORD   0
  123
  124
  125
  126                                   ;KT11-D STATUS REGISTER ADDRESSES
  127
  128  000102  177572                  SR0:    177572
  129  000104  177574                  SR1:    177574
  130  000106  177576                  SR2:    177576
  131  000110  172516                  SR3:    172516
  132
  133
  134
  135                                   ;KERNAL PAGE DESCRIPTOR REGISTERS
  136
  137  000112  172300                  KIPDR0: 172300
  138  000114  172302                  KIPDR1: 172302
  139  000116  172304                  KIPDR2: 172304
  140  000120  172306                  KIPDR3: 172306
  141  000122  172310                  KIPDR4: 172310
  142  000124  172312                  KIPDR5: 172312
  143  000126  172314                  KIPDR6: 172314
  144  000130  172316                  KIPDR7: 172316
  145
  146                                   ;KERNAL PAGE ADDRESS REGISTERS
  147
  148  000132  172340                  KIPAR0: 172340
  149  000134  172342                  KIPAR1: 172342
  150  000136  172344                  KIPAR2: 172344
  151  000140  172346                  KIPAR3: 172346
  152  000142  172350                  KIPAR4: 172350
  153  000144  172352                  KIPAR5: 172352
  154  000146  172354                  KIPAR6: 172354
  155  000150  172356                  KIPAR7: 172356
  156
  157                                   ;KT11 VECTOR ADDRESS
  158
  159  000152  000250  000252          SEGVEC: 250,252
```

```
                000200                          .=200
000200  000167  003250                          JMP     BEGN11          ;JUMP TO START TO TEST ALL OF
                                                                        ;MEMORY WITHOUT CHOOSING LIMITS
000204  000167  002536                          JMP     BEGIN1          ;JUMP TO START CHOOSE LIMITS
000210  000167  003666                          JMP     BEGIN3          ;JUMP TO RESTART USING LAST SPECIFIED LIMITS


000214  000000                  IN2ADI: 0
000216  000000                  FRSTADI: 0
```



```
                000300                          .=300


000300  016701  002344          RESTOR: MOV     TO1,R1
000304  016702  005022                  MOV     FROM,R2
000310  016703  005014                  MOV     LENGTH,R3
000314  012122                  1$:     MOV     (R1)+,(R2)+
000316  077302                          SOB     R3,1$
000320  000137  002004                  JMP     @#204


                                ;*******************************************************************


000324  010704                  LOADER: MOV     PC,R4
000326  062704  011164                  ADD     #LAST+2-.,R4
000332  016702  002334                  MOV     LODER,R2
000336  012701  003132                  MOV     #17774-LAST/2,R1
000342  012442                  1$:     MOV     (R4)+,-(R2)
000344  077102                          SOB     R1,1$
000346  004767  006666                  JSR     PC,TYPE
000352  011466                          FIN
000354  000000                          HALT
```

```
                                        ;*****************************************************************************
                    001130                      .=1100
                                        ;*****************************************************************************
                                        ;ROUTINE TO TYPE ASCII 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: $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
                                        ;CALL:
                                        ;       JSR     PC,$TYPE                ;CALL TYPE ROUTINE
                                        ;       MESADDR                         ;FIRST ADRESS OF MESSAGE
001100  177564                STPS:   177564                          ;TTY PRINTER STATUS REG. ADDRESS
001102  177566                STPB:   177566                          ;TTY PRINTER BUFFER REG. ADDRESS
001104  000                 $NULL:  .BYTE   0                       ;CONTAINS NULL CHARACTER FOR FILLS
001105  002                 $FILLS: .BYTE   2                       ;CONTAINS # OF FILLER CHARACTERS REQUIRED
001106  000                 STPFLG: .BYTE   0                       ;"TERMINAL AVAILABLE" FLAG (0=YES)
001107  000                         .BYTE   0                       ;RESERVED

001110  105767  177772      $TYPE:  TSTB    STPFLG                  ;IS THERE A TERMINAL?
001114  001402                      BEQ     6$                      ;BR IF YES
001116  000000                      HALT                            ;HALT HERE IF NO TERMINAL
001120  000407                      BR      7$                      ;LEAVE
001122  010046              6$:     MOV     R0,-(SP)                ;SAVE R0
001124  017600  000002              MOV     @2(SP),R0               ;GET ADDRESS OF ASCII STRING
001130  112046              1$:     MOVB    (R0)+,-(SP)             ;PUSH CHARACTER TO BE TYPED ONTO STACK
001132  001005                      BNE     2$                      ;BR IF IT ISN'T THE TERMINATOR
001134  005726                      TST     (SP)+                   ;IF TERMINATOR POP IT OFF THE STACK
001136  012600                      MOV     (SP)+,R0                ;RESTORE R0
001140  062716  000002      7$:     ADD     #2,(SP)                 ;ADJUST RETURN PC
001144  000207                      RTS     PC                      ;RETURN
001146  004767  000026      2$:     JSR     PC,5$                   ;GO TYPE THIS CHARACTER
001152  122726  000012      3$:     CMPB    #12,(SP)+               ;CHECK IF THE CHAR. TYPED WAS A  LINE FEED
001156  001364                      BNE     1$                      ;GO GET NEXT CHAR, IF NOT LINE FEED
001160  016746  177720              MOV     $NULL,-(SP)             ;GET # OF FILLER CHARS. NEEDED
                                                                    ;AND THE NULL CHAR,
001164  105366  000001      4$:     DECB    1(SP)                   ;DOES A NULL NEED TO BE TYPED?
001170  002770                      BLT     3$                      ;BR IF NO--GO POP THE NULL OFF OF STACK
001172  004767  000002              JSR     PC,5$                   ;GO TYPE A NULL
001176  000772                      BR      4$                      ;LOOP
001200  105777  177674      5$:     TSTB    @STPS                   ;WAIT UNTIL PRINTER IS READY
001204  100375                      BPL     5$
001206  116677  000002  177666      MOVB    2(SP),@STPB             ;LOAD CHAR TO BE TYPED INTO DATA REG.
001214  000207                      RTS     PC
001216  000062                      .BLKB   62                      ;RESERVE SOME MORE CORE FOR OVERLAY CAPIBILITIES


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



```
                                        ;ROUTINE TO SIZE MEMORY
                                        ;CALL:
                                        ;       JSR     PC,$SIZE
                                        ;       RETURN
                                        ;SLSTAD WILL CONTAIN THE LAST VIRTUAL ADDRESS OF THE LAST BANK
                                        ;SLSTBK WILL CONTAIN THE LAST BANK AS A SAF
                                        ;SKT11 IS THE MEMORY MANAGEMENT KEY
                                        ;BIT07 = 0 DON'T USE MEMORY MANAGEMENT
                                        ;BIT15 = 0 DON'T HAVE MEMORY MANAGEMENT OPTION

001300  010546              $SIZE:  MOV     R5,-(SP)                ;SAVE R5 ON THE STACK
001302  010446                      MOV     R4,-(SP)
001304  010346                      MOV     R3,-(SP)
001306  010146                      MOV     R1,-(SP)                ;SAVE R1 ON THE STACK
001310  010046                      MOV     R0,-(SP)                ;SAVE R0 ON THE STACK
001312  010605                      MOV     SP,R5                   ;SAVE THE STACK POINTER
001314  012701  003776              MOV     #3776,R1                ;
001320  105727                      TSTB    (PC)+                   ;USE MEMORY MANAGEMENT?
001322  000000              $KT11:  0
001324  100152                      BPL     $CORE                   ;BR IF NO
001326  010746              $SEG:   MOV     PC,-(SP)
001330  062716  000322              ADD     #$CORE-.,(SP)
001334  012637  000004              MOV     (SP)+,@#ERRVEC          ;SET FOR TIMEOUT
001340  005777  176536              TST     @SR0                    ;KT11 ARE YOU THERE?
001344  052767  100000  177750      BIS     #BIT15,$KT11            ;YES--SET KT11 KEY
001352  012703  077406              MOV     #77406,R3
001356  010377  176536              MOV     R3,@KIPDR0
001362  010377  176526              MOV     R3,@KIPDR1
001366  010377  176524              MOV     R3,@KIPDR2
001372  010377  176522              MOV     R3,@KIPDR3
001376  010377  176520              MOV     R3,@KIPDR4
001402  010377  176516              MOV     R3,@KIPDR5
001406  010377  176514              MOV     R3,@KIPDR6
001412  010377  176512              MOV     R3,@KIPDR7

001416  005077  176518              CLR     @KIPAR0
001422  012777  000200  176504      MOV     #200,@KIPAR1
001430  012777  000400  176500      MOV     #400,@KIPAR2
001436  012777  000600  176474      MOV     #600,@KIPAR3
001444  012777  001000  176470      MOV     #1000,@KIPAR4
001452  012777  001200  176464      MOV     #1200,@KIPAR5
001460  012777  001400  176460      MOV     #1400,@KIPAR6
001466  012777  007600  176454      MOV     #7600,@KIPAR7

001474  010703                      MOV     PC,R3
001476  000241                      CLC
001500  006103                      ROL     R3
001502  006103                      ROL     R3
001504  006103                      ROL     R3
001506  006103                      ROL     R3
001510  042703  177770              BIC     #177770,R3
```

```
   311  001514  022703  000006                  CMP     #6,R3
   312  001520  001403                          BEQ     2$
   313
   314
   315  001522  016704  176420                  MOV     KIPAR6,R4
   316  001526  000425                          BR      3$
   317
   318
   319
   320  001530  016704  176410          2$:     MOV     KIPAR5,R4
   321  001534  012767  123776  000070          MOV     #123776,1$+2
   322  001542  012703  120000                  MOV     #120000,R3
   323  001546  010367  003354                  MOV     R3,SEG0+2
   324  001552  010367  003356                  MOV     R3,SEG0+10
   325  001556  010367  003376                  MOV     R3,SEG10+2
   326  001562  010367  003400                  MOV     R3,SEG10+10
   327  001566  012767  137776  003346          MOV     #137776,SEG0+16
   328  001574  012767  137776  003372          MOV     #137776,SEG10+16
   329
   330
   331
   332  001602  010467  001060          3$:     MOV     R4,PAGE
   333  001606  005014                          CLR     (R4)
   334
   335
   336
   337
   338  001610  012777  000001  176264          MOV     #1,@$R0                 ;TURN ON MEMORY MANAGEMENT
   339  001616  010746                          MOV     PC,-(SP)
   340  001620  062716  000026                  ADD     #$KTOUT-.,(SP)
   341  001624  012637  000004                  MOV     (SP)+,@#ERRVEC  ;SET FOR TIME OUT
   342  001630  005737  143776          1$:     TST     @#143776                ;TRAP ON NON-EX-MEM
   343  001634  062714  000040                  ADD     #40,(R4)                ;MAKE A 1K STEP
   344  001640  027714  176304                  CMP     @KIPAR7,(R4)            ;LAST ONE?
   345  001644  003371                          BGT     1$                      ;NO--TRY IT
   346  001646  011400                  $KTOUT: MOV     (R4),R0         ;GET LAST BANK+1
   347  001650  000420                          BR      $SIZEX
   348  001652  010746                  $CORE:  MOV     PC,-(SP)
   349  001654  062716  000032                  ADD     #$COROUT-.,(SP)
   350  001660  012637  000004                  MOV     (SP)+,@#ERRVEC  ;SET FOR TIMEOUT
   351  001664  005000                          CLR     R0                      ;SET UP BANK
   352  001666  062701  004000          1$:     ADD     #4000,R1                ;INCREMENT BY 1K
   353  001672  062700  000040                  ADD     #40,R0                  ;1K STEP
   354  001676  005711                          TST     (R1)                    ;TRAP ON TIME OUT
   355  001700  022701  177776                  CMP     #177776,R1              ;LAST ONE
   356  001704  001370                          BNE     1$                      ;NO--TRY AGAIN
   357  001706  162701  004000          $COROUT:        SUB     #4000,R1        ;
   358  001712  162700  000040          $SIZEX: SUB     #40,R0                  ;DROP BACK
   359  001716  012737  000006  000004          MOV     #6,@#ERRVEC             ;SET FOR ERRORS
   360  001724  010506                          MOV     R5,SP                   ;RESTORE THE STACK
   361  001726  010167  000032                  MOV     R1,$LSTAD               ;
   362  001732  010067  000030                  MOV     R0,$LSTBK               ;
   363  001736  005767  177340                  TST     $KT11                   ;IS MEMORY MANAGEMENT THERE
   364  001742  100002                          BPL     1$                      ;IF NOT BRANCH
```



```
   365  001744  005377  176132                  CLR     @$R0                    ;DISABLE MEMORY MANAGEMENT
   366  001750  012600                  1$:     MOV     (SP)+,R0                ;RESTORE R0
   367  001752  012601                          MOV     (SP)+,R1                ;RESTORE R1
   368  001754  012603                          MOV     (SP)+,R3
   369  001756  012604                          MOV     (SP)+,R4
   370  001760  012605                          MOV     (SP)+,R5                ;RESTORE R5
   371  001762  000207                          RTS     PC
   372  001764  000000          $LSTAD: .WORD   0                       ;CONTAINS THE LAST ADDRESS
   373  001766  000000          $LSTBK: .WORD   0                       ;CONTAINS THE LAST BANK
```

```
                                        ;ROUTINE TO GET TYPED ASCII DATA AND CONVERT TO OCTAL.
                                        ;LEAVES THE OCTAL DATA IN THE ADDRESS FOLLOWING THE CALL.
                                        ;CALL: RECD             ;(A TRAP TYPE INST.)
                                        ;       .WORD   0       ;CONTAINS RETURNED DATA
                                        ;       CONTINUE        ;RETURNS HERE

001770  177560                  TKS:    177560
001772  177562                  TKB:    177562

001774  000240          .RECD:  NOP
001776  005046                  CLR     -(SP)
002000  105777  177764          1$:     TSTB    @TKS            ;WAIT FOR USER RESPONSE
002004  100375                  BPL     1$
002006  117746  177760                  MOVB    @TKB,-(SP)      ;GET ASCII CHARACTER
002012  042716  177600                  BIC     #177600,(SP)    ;
002016  122716  000177                  CMPB    #177,(SP)       ;CHECK IF RUBOUT
002022  001013                  BNE     2$              ;BRANCH IF NOT RUBOUT
002024  012767  000134  000142          MOV     #'\,    4$
002032  004767  005202                  JSR     PC, TYPE                ;ECHO BACK SLASH
002036  002174                  4$
002040  005726                  4$:     TST     (SP)+           ;POP LAST TYPED CHAR OFF THE STACK
002042  006016                  ROR     (SP)            ;SHIFT LAST TYPD CHARACTER
002044  006216                  ASR     (SP)
002046  006216                  ASR     (SP)
002050  000753                  BR      1$              ;AND WAIT FOR NEXT CHAR.

002052  122716  000015          2$:     CMPB    #15,(SP)        ;CHECK IF CARRIAGE RETURN
002056  001011                  BNE     3$              ;BRANCH IF NOT A CARRIAGE RETURN
002060  004767  005154                  JSR     PC,TYPE
002064  002647                  $CRLF
002066  005726                  TST     (SP)+           ;POP CARRIAGE RETURN OFF THE STACK
002070  012676  000000                  MOV     (SP)+,@(SP)     ;PUT DATA IN ADDRESS FOLLOWING CALL
002074  062716  000002                  ADD     #2,(SP)
002100  000207                  RTS     PC
002102  111667  000066          3$:     MOVB    (SP),4$

002106  004767  005126                  JSR     PC,TYPE         ;GO TO PRINT ROUTINE
002112  002174                  4$                              ;ECHO CHARACTER
002114  042767  177707  000052          BIC     #177707,4$
002122  022767  000060  000044          CMP     #60,4$
002130  001407                  BEQ     5$
002132  004767  005102                  JSR     PC,TYPE
002136  002647                  $CRLF
002140  004767  005074                  JSR     PC,TYPE
002144  011217                  SORRY
002146  000734                  BR      6$
002150  042716  177770          5$:     BIC     #177770,(SP)    ;SAVE ONLY 3 MSBS
002154  006366  000002                  ASL     2(SP)           ;SHIFT LAST CHARACTER OVER
002160  006366  000002                  ASL     2(SP)           ;THREE PLACES
002164  006366  000002                  ASL     2(SP)
002170  052616                  6$:     BIS     (SP)+,(SP)      ;INSET OCTAL CHAR
```



```
002172  000702                  BR      1$              ;GO WAIT FOR NEXT CHAR.
002174  000000          4$:     .WORD   0               ;CONTAINS CHARACTER TO BE TYPED
```

```
                                ;ROUTINE TO SET PARITY ENABLE ON MA/MF PARITY MEMORIES
            172100              PARCSR= 172100  ;ADDRESS OF FIRST POSSIBLE PARITY REG.


002176  012737  002254  000114  MAMF1   MOV     #PARSRV,@#PARVEC        ;LOAD VECTOR
002204  012737  000340  000116          MOV     #340,@#PARVEC+2         ;AND PRIORITY LEVEL
002212  012737  000006  000004          MOV     #ERRVEC+2,@#ERRVEC      ;SET TIME OUT TRAP VECTOR
002220  012737  000002  000006          MOV     #RTI,@#ERRVEC+2         ;DO RTI ON TIMEOUT TRAP
002226  012700  172100                  MOV     #PARCSR,R0              ;GET FIRST POSSIBLE ADDRESS
002232  012702  000001                  MOV     #1,R2                   ;SET REG. COUNTER

002236  012720  000001          1$:     MOV     #1,(R0)+                ;SET ACTION ENABLE IF THERE
002242  006302                          ASL     R2                      ;CHECK IF 16 REG, HAVE BEEN DONE
002244  103374                          BCC     1$                      ;REPEAT IF 16 NOT ENABLED
002246  005037  000006                  CLR     @#ERRVEC+2              ;RESTORE HALT ON TIMEOUT
002252  000207                          RTS     PC                      ;RETURN


```



```
                                ;PARITY ERROR SERVICE ROUTINE
002254  004767  004760          PARSRV: JSR     PC,     TYPE    ;TYPE PARITY ERROR SO PARITY
002260  011131                          PARERR                  ;ERROR TEST STARTED
002262  010746                          MOV     PC,     -(SP)   ;MAKE 2S POSITION INDEPENDENT
002264  062716  000120                  ADD     #2$-.,  (SP)    ;
002270  012637  000114                  MOV     (SP)+,@#PARVEC  ;SET PARITY ERROR TRAP
002274  010746                          MOV     PC,     -(SP)   ;MAKE 4S POSITION INDEPENDENT
002276  062716  000216                  ADD     #4$-.,  (SP)    ;
002302  012637  000004                  MOV     (SP)+,  @#ERRVEC
002306  010746                          MOV     PC,     -(SP)   ;MAKE 5S POSITION INDEPENDENT
002310  062716  000214                  ADD     #5$-.,  (SP)    ;
002314  012637  000250                  MOV     (SP)+,@#MMVEC   ;SET MEMORY MANAGEMENT ABORT TRAP
002320  005000                          CLR     R0              ;
002322  005767  176774                  TST     SKT11           ;IS MEM MANG, THERE?
002326  100024                          BPL     1$              ;IF NOT BRANCH
002330  012702  077406                  MOV     #77406,R2       ;SET UP MEM MANGT.
002334  005077  175572                  CLR     @KIPAR0
002340  010277  175546                  MOV     R2,@KIPDR0
002344  005077  175564                  CLR     @KIPAR1
002350  005077  175542                  CLR     @KIPDR2
002354  010277  175534                  MOV     R2,@KIPDR1
002360  012777  007600  175562          MOV     #7600,@KIPAR7
002366  010277  175536                  MOV     R2,@KIPDR7
002372  012777  000001  175502          MOV     #1,@SR0
                                                                ;ENABLE MEM MANGT.


002400  005720                  1$:     TST     (R0)+           ;SCAN MEMORY FOR PARITY
002402  000776                          BR      1$


002404  004767  004630          2$:     JSR     PC,TYPE         ;TYPE LOCATION GIVING PARITY ERROR
002410  011021                          PAERR
002412  017777  175516  175522          MOV     @KIPAR1,@KIPAR4
002420  014046                          MOV     -(R0),-(SP)
002422  004067  004672                  JSR     R0,@@2OCT
002426     006                          ,BYTE   6
002427     003                          ,BYTE   3
002430  005767  176666          6$:     TST     SKT11
002434  100005                          BPL     3$
002436  005077  175448                  CLR     @SR0
002442  004767  004572                  JSR     PC,TYPE
002446  002647                          $CRLF
002450  000005                  3$:     RESET
002452  010746                          MOV     PC,-(SP)
002454  062716  177600                  ADD     #PAR0RV-.,(SP)
002460  012637  000114                  MOV     (SP)+,@#PARVEC
002464  010746                          MOV     PC,-(SP)
002466  062716  175320                  ADD     #ERRVEC+2-.,(SP)
002472  012637  000004                  MOV     (SP)+,@#ERRVEC  ;RESET PARITY ERROR TRAP AND ERROR VECTOR
002476  012737  000252  000250          MOV     #MMVEC+2,@#MMVEC        ;RESET MEM MANGT. ABORT TRAP
002504  010746                          MOV     PC,-(SP)
002506  062716  001374                  ADD     #BEGIN3-.,(SP)
002512  000136                          JMP     @(SP)+          ;JUMP RESTART
```

```
                                                                                
                                                                                
                                                                                
002514  004767  004520          4$:     JSR     PC,     TYPE
002520  011061                          NOPARE
002522  000742                          BR      6$
                                                                                
                                                                                
                                        ;MEMORY MANAGEMENT ABORT ROUTINE
002524  062777  000200  175402  5$:     ADD     #200,@K!PAR1    ;ADD NEXT 4K TO ADDRESS
002532  012700  020000                  MOV     #20000,R0       ;RESET VIRTUAL ADDRESS
002536  000002                          RTI
```



```
                                        ;*****************************************************************
                                        ;COMMON TAGS
                                                                                
                                                                                
002540  000000                  $PASS!  ,WORD   0               ;CONTAINS PASS COUNT
002542  000000                  $TSTNM: ,WORD   0               ;CONTAINS THE TEST NUMBER
002544  000000                  $ICNT!  ,WORD   0               ;CONTAINS SUBTEST ITERATION COUNT
002546  000000                  $LPADR! ,WORD   0               ;CONTAINS SCOPE LOOP ADDRESS
002550  000000                  $LPERR! ,WORD   0               ;CONTAINS SCOPE RETURN FOR ERRORS
002552  000000                  $ERTTL! ,WORD   0               ;CONTAINS TOTAL ERRORS DETECTED
002554     000                  $ERFLG! ,BYTE   0               ;CONTAINS ERROR FLAG
002555     000                          ,BYTE   0               ;RESERVED--NOT TO BE USED
002556  000000  000000                  ,WORD   0,0             ;RESERVED--NOT TO BE USED
002562     000                  $ITEMB! ,BYTE   0               ;CONTAINS ITEM CONTROL BYTE
002563     000                          ,BYTE   0               ;RESERVED--NOT TO BE USED
002564  000000                  $HLTAD! ,WORD   0               ;CONTAINS PC OF LAST HLT INSTRUCTION
002566  000000                  $GDADR! ,WORD   0               ;CONTAINS ADDRESS OF 'GOOD' DATA
002570  000000                  $BDADR! ,WORD   0               ;CONTAINS ADDRESS OF 'BAD' DATA
002572  000000                  $GDDAT! ,WORD   0               ;CONTAINS 'GOOD' DATA
002574  000000                  $BDDAT! ,WORD   0               ;CONTAINS 'BAD' DATA
002576  000000                  STMP0!  ,WORD   0               ;USER DEFINED
002600  000000                  STMP1!  ,WORD   0               ;USER DEFINED
002602  000000                  STMP2!  ,WORD   0               ;USER DEFINED
002604  000000                  TSTADR! ,WORD   0               ;TEST ADDRESS
002606  000000                  LOLMT!  ,WORD   0               ;LOW LIMIT
002610  017776                  HILMT!  ,WORD   17776           ;HIGH LIMIT
002612  000000                  MOD!    ,WORD   0               ;MODULE NUMBER
002614  000000                  COL!    ,WORD   0               ;COLUMN NUMBER
002616  000000                  ROW!    ,WORD   0               ;ROW NUMBER
002620  000000                  PCHRG!  ,WORD   0               ;PRECHARGE
002622  023420                  CYCNT!  ,WORD   10000.          ;CYCLE COUNT
002624  177777                  TSTPAT! ,WORD   -1              ;TEST PATTERN
002626  177777                  BCKGRD! ,WORD   -1              ;BACK GROUND PATTERN
002630  000000                  INSTAD! ,WORD   0               ;ADDRESS OF INSTRUCTION UNDER TEST
002632  000400                  MSKBIT! ,WORD   BIT08           ;MASK BIT
002634  000000                  SV40$!  ,WORD   0
002636  000000                  $VIN1!  ,WORD   0
002640  000000                  $VIN2!  ,WORD   0
002642  000000                  ESCAPE! ,WORD   0
002644  000207                  $BELL!  ,ASCIZ  <207>           ;CODE FOR BELL
002646     077                  $QUES!  ,ASCII  /?/             ;QUESTION MARK
002647     015                  $CRLF!  ,ASCII  <15>            ;CARRIAGE RETURN
002650  000012                  $LF!    ,ASCIZ  <12>            ;LINE FEED
002652  000000                  PASFLG! ,WORD   0
002654  000000                  $ELF!   ,WORD   0
002656  177777                  REDONE! ,WORD   -1
002660  000000                  LOBASE! ,WORD   0
002662  000000                  HIBASE! ,WORD   0
002664  000000                  LSTBK!  ,WORD   0
002666  000000                  PAGE!   ,WORD   0
002670  000000                  TO1!    ,WORD   0               ;RELOCATE TO HERE IF 0 CHOSEN
002672  000000                  LODER!  ,WORD   0               ;LOADER IS HERE
```

```
                                ;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

002674                          $ERRTB: ;ITEM 1
                                        EM1             ;TEST PATTERN
002674  007614                          DH1             ;ERROR  FAILING
                                                        ;PC     ADDRESS EXPECT  RCV'D   COL     ROW     PRECHARGE
002700  010270                          DT1             ;$HLTAD $BDADR  $GDDAT  $BDDAT  $TMP0   $TMP1   $TMP2
002702  010310                          DF1             ;ALL NUMBERS ARE TYPED IN OCTAL

                                ;ITEM 2
002704  007633                          EM2             ;INSTRUCTION CHANGED
002706  010016                          DH2             ;ERROR  INST    EXPECT  RCV'D,
                                                        ;PC     ADDRESS DATA    DATA    COL     ROW     PRECHARGE
002710  010270                          DT1             ;$HLTAD $BDADR  $GDDAT  $BDDAT  $TMP0   $TMP1   $TMP2
002712  010310                          DF1             ;ALL NUMBERS ARE TYPED IN OCTAL

                                ;ITEM 3
002714  007661                          EM3             ;BACK GROUND CHANGED
002716  010143                          DH3             ;ERROR  BCKGRD  EXPECT  RCV'D
                                                        ;PC     ADDRESS DATA    DATA    COL     ROW     PRECHARGE
002720  010270                          DT1             ;$HLTAD $BDADR  $GDDAT  $BDDAT  $TMP0   $TMP1   $TMP2
002722  010310                          DF1             ;ALL NUMBERS ARE TYPED IN OCTAL
                                ;*************************************************************************
```



```
                                ;*************************************************************************
                                ;THIS IS WHERE THE TEST INSTRUCTIONS ARE STORED
002724  005010                  INSTR1: CLR     (R0)
002726  077102                  INSTR2: SOB     R1,INSTR1
002730  000207                  INSTR3: RTS     PC
                                ;END OF TEST INSTRUCTIONS
002732  005010                  INSIDE: CLR     (R0)
002734  077102                          SOB     R1,INSIDE
002736  000207                          RTS     PC

002740  000000  177777  177777  OUTSID: .WORD   0,-1,-1
```

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

                                        ;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

002746  012706                 BEGIN1: MOV     PC,     SP      ;SET STACK POINTER AND
002750  002766  176130                  ADD     #1100.,SP       ;MAKE IT POSITION INDEPENDANT
002754  005067  177672                  CLR     PASFLG          ;PASFLG IS COUNTER FOR WHEN
                                                                ;BOTH PICKS & DROPS
002760  004767  004254                  JSR     PC,     TYPE
                                        2
002764  012767  000200  176326          MOVLF   #BIT07,SKT11    ;USE MEM MANG. IF THERE
002774  004767  176300                  JSR     PC,     SSIZE   ;SIZE MEM
003000  004767  004234                  JSR     PC,     TYPE    ;TYPE "LAST ADDRESS IN MEMORY"
003004  010771                          MEMEND                  ;
003006  005767  176310                  TST     SKT11           ;IS MEMORY MANAGEMENT THERE?
003012  100006                          BPL     2$              ;BRANCH IF NO MEM MANAGEMENT
003014  016777  176746  177644          MOV     SLSTBK, @PAGE   ;PAGE CONTAINS KIPAR6 OR KIPAR5
003022  012777  000001  175052          MOV     #1,     @SR0    ;TURN ON MEM. MANAGEMENT
003030  016746  176730          2$:     MOV     SLSTADR,-(SP)   ;GET READY TO TYPE ADDRESS IN OCTAL
003034  004067  004260                  JSR     R0,$B20CT
003040     006                          .BYTE   6
003041     003                          .BYTE   3               ;PHYSICAL ADDRESS IF MEM MANG. THERE
003042  005767  176254                  TST     SKT11           ;
003046  100002                          BPL     3$              ;
003050  005077  175026                  CLR     @SR0            ;
003054  004767  004160          3$:     JSR     PC,     TYPE    ;
003060  002647                          $CRLF
003062  004767  004152                  JSR     PC,     TYPE    ;GO TO TYPE "TYPE STARTING
                                                                ;BANK NUMBER AND CR
003066  010534                  1$:     STARTB                  ;REFER TO SECTION 4.3.1 BEFORE CHOOSING"
003070  012767  010653  177770          MOV     #START2,1$
003076  004767  176672                  JSR     PC,     .RECD   ;GET STARTING BANK NUMBER
003102  000000          STBANK: .WORD   0                       ;AND LEAVE IT HERE
003104  004767  004130                  JSR     PC,     TYPE    ;GO TO TYPE "TYPE NUMBER OF 4K
003110  010674                  1$:     NUMBER                  ;BANK$ TO BE TESTED AND CR"
003112  012767  010753  177770          MOV     #NUMB2, 1$      ;
003120  004767  176650                  JSR     PC,     .RECD   ;GET NUMBER OF BANKS
003124  000000          BANKS:  .WORD   0                       ;AND LEAVE IT HERE
003126  005067  177522                  CLR     SELF
003132  012767  024042  002166          MOV     #BEGIN2+20000, BEG2     ;THESE FOUR LINES
003140  012767  007736  002162          MOV     #7736, LENGTH   ;TO SET UP FOR
003146  012767  000100  002156          MOV     #FIRST, FROM    ;NO LIMIT CHOOSING ON RESTART
003154  012767  020100  002152          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
```



```


003162  016704  176600                  MOV     SLSTBK,R4
003166  162704  000140                  SUB     #140,R4
003172  022704  001400                  CMP     #1400,R4
003176  101002                          BHI     1$
003200  012704  001400                  MOV     #1400,R4
003204  000241          1$:     CLC
003206  006104                          ROL     R4
003210  006104                          ROL     R4
003212  006104                          ROL     R4
003214  006104                          ROL     R4
003216  006104                          ROL     R4
003220  006104                          ROL     R4
003222  010402                          MOV     R4,R2
003224  005767  177442                  TST     LODER
003230  001027                          BNE     9$              ;BRANCH IF SOMETHING IN LODER
                                                                ;THAT IS LOADER ALREADY PULLED IN
003232  010403                          MOV     R4,R3
003234  062703  020000                  ADD     #20000,R3
003240  010367  177426                  MOV     R3,LODER
003244  010705                          MOV     PC,R5   ;TO
003246  062705  006244                  ADD     #LAST+2-.,R5
003252  012700  003132                  MOV     #17774-LAST/2,R0;LENGTH
003256  014325          2$:     MOV     -(R3),(R5)+
003260  077002                          SOB     R0,2$
003262  004767  003752                  JSR     PC,TYPE
003266  011402                          RELOD
003270  012746  000324                  MOV     #LOADER,-(SP)
003274  004067  004020                  JSR     R0,$B20CT
003300     006                          .BYTE   6
003301     002                          .BYTE   2
003302  004767  003732                  JSR     PC,TYPE
003306  011434                          RELOD2
003310  005767  177566          9$:     TST     STBANK
003314  001056                          BNE     4$
003316  016746                          MOV     PC,-(SP)
003320  042726  017777                  BIC     #17777,(SP)+
003324  001052                          BNE     4$
003326  010406                          MOV     R4,@P
003330  062706  001100                  ADD     #1100,SP
003334  062704  000100                  ADD     #FIRST,R4
003340  016467  177324                  MOV     R4,TO1
003344  016700  001760                  MOV     LENGTH,R0
003350  016701  001756                  MOV     FROM,R1
003354  012124          3$:     MOV     (R1)+,(R4)+
003356  077002                          SOB     R0,3$
003360  062702  004042                  ADD     #BEGIN0,R2
003364  004767  003650                  JSR     PC,TYPE
003370  011264                          PRO70
003372  016746  177272                  MOV     TO1,-(SP)
003376  062716  000104                  ADD     #104,(SP)
003402  004067  003712                  JSR     R0,$B20CT
003406     006                          .BYTE   6
```

```
   722  003457     002                          .BYTE   2
   723  003460  004767  003624                  JSR     PC,TYPE
   724  003464  002647                          $CRLF
   725  003466  004767  003616                  JSR     PC,TYPE
   726  003472  011332                          PROBAK
   727  003474  016746  177240                  MOV     @01,-(SP)
   728  003500  062716  000200                  ADD     #RESTOR-100,(SP)
   729  003504  004067  003660                  JSR     R0,$B20CT
   730  003510     006                          .BYTE   6
   731  003511     002                          .BYTE   2
   732  003512  004767  003572                  JSR     PC,TYPE
   733  003516  002647                          $CRLF
   734  003520  000112                          JMP     (R2)
   735  003522  000573                  4$:     BR      BEGIN2
   736
   737                                          ;THIS IS THE STARTING POINT IF NO LIMITS ARE TO
   738                                          ;BE SET AND ALL OF EXISTING MEMORY IS TO BE TESTED
   739
   740
   741  003524  010706                  BEGN11: MOV     PC,     SP      ;SETUP STACK POINTER AND
   742  003526  062706  175422                  ADD     #1100.,,SP      ;MAKE IT POSITION INDEPENDANT
   743  003532  005067  177164                  CLR     PASFLG          ;PASFLG IS COUNTER WHEN BOTH PICKS
   744                                                                  ;AND DROPS ARE USED
   745  003536  012767  000200  175626          MOV     #BIT07, $KT11   ;BIT 7 SAYS SEGMENTATION IS THERE
   746  003544  004767  175600                  JSR     PC,     $SIZE   ;SIZE MEMORY
   747  003550  004767  003534                  JSR     PC,     TYPE    ;
   748  003554  010771                          MEMEND                  ;
   749  003556  005767  175610                  TST     $KT11           ;
   750  003562  100006                          BPL     2$              ;
   751  003564  016777  176246  177144          MOV     $LSTBK, @PAGE   ;
   752  003572  012777  000001  174352          MOV     #1,     @SR0    ;
   753  003600  016746  176230          2$:     MOV     $LSTADR,-(SP)   ;
   754  003604  004067  003560                  JSR     R0,$B20CT
   755  003610     006                          .BYTE   6
   756  003611     003                          .BYTE   3
   757  003612  005767  175554                  TST     $KT11           ;
   758  003616  100002                          BPL     3$              ;
   759  003620  005077  174326                  CLR     @SR0            ;
   760  003624  004767  003440          3$:     JSR     PC,     TYPE    ;
   761  003630  002647                          $CRLF
   762  003632  012767  000001  177312          MOV     #1,     STBANK  ;STARTING BANK # = 1 LOC = 20000
   763  003640  016767  176172  177066          MOV     $LSTBK, LSTBK   ;LAST BANK IS STORED IN LSTBK
   764  003646  062767  000040  177060          ADD     #40,LSTBK       ;
   765  003654  006067  177054                  ROR     LSTBK           ;DIVIDE LSTBK BY 200
   766  003660  006067  177050                  ROR     LSTBK           ;TO GIVE NUMBER OF
   767  003664  006067  177044                  ROR     LSTBK           ;BANKS PRESENT IN
   768  003670  006067  177040                  ROR     LSTBK           ;EXISTING MEMORY
   769  003674  006067  177034                  ROR     LSTBK           ;
   770  003700  006067  177030                  ROR     LSTBK           ;
   771  003704  006067  177024                  ROR     LSTBK           ;
   772  003710  005367  177020                  DEC     LSTBK           ;NOW DECREASE ONE FOR BANKS
   773  003714  016767  177014  177252          MOV     LSTBK,  BANKS   ;TO BE TESTED AND STORE
   774  003722  012767  177777  176774          MOV     #-1,    SELF    ;1 = RELOCATION REQUIRED
   775  003730  012767  024042  001440          MOV     #BEGIN2+20000, BEG2
```



```
   776  003736  012767  007736  001434          MOV     #7736,LENGTH
   777  003744  012767  000100  001430          MOV     #FIRST, FROM
   778  003752  012767  020100  001424          MOV     #FIRST+20000, TO
   779  003760  005067  176742                  CLR     REDONE          ;RELOCATION EXIT COUNTER
   780  003764  005767  176752                  TST     LODER
   781  003770  001050                          BNE     BEGIN2          ;BRANCH IF SOMETHING IN LODER
   782                                                                  ;THAT IS LOADER ALREADY PULLED IN
   783  003772  016704  176040                  MOV     $LSTBK,R4
   784  003776  162704  000140                  SUB     #140,R4
   785  004002  022704  001400                  CMP     #1400,R4
   786  004006  101002                          BHI     1$
   787  004010  012704  001400                  MOV     #1400,R4
   788  004014  000241                  1$:     CLC
   789  004016  006104                          ROL     R4
   790  004020  006104                          ROL     R4
   791  004022  006104                          ROL     R4
   792  004024  006104                          ROL     R4
   793  004026  006104                          ROL     R4
   794  004030  006104                          ROL     R4
   795  004032  010402                          MOV     R4,R2
   796  004034  010403                          MOV     R4,R3
   797  004036  062703  020000                  ADD     #20000,R3
   798  004042  010367  176674                  MOV     R3,LODER
   799  004046  010705                          MOV     PC,R5   ;TO
   800  004050  062705  005512                  ADD     #LAST+2-.,R5
   801  004054  012700  003132                  MOV     #17774-LAST/2,R0;LENGTH
   802  004060  014325                  4$:     MOV     -(R3),(R5)+
   803  004062  077002                          SOB     R0,4$
   804  004064  004767  003220                  JSR     PC,TYPE
   805  004070  011402                          RELOD
   806  004072  012746  000324                  MOV     #LOADER,-(SP)
   807  004076  004067  003266                  JSR     R0,$B20CT
   808  004102     006                          .BYTE   6
   809  004103     002                          .BYTE   2
   810  004104  004767  003260                  JSR     PC,TYPE
   811  004110  011434                          RELOD2
   812
   813                                          ;THE CONSOLE SWITCHES ARE TO BE SET.  LOOK AT SECTION
   814                                          ;4.1 AT THE FRONT OF THIS DOC
   815
   816  004112  004767  003172          BEGIN2: JSR     PC,     TYPE
   817  004116  002647                          $CRLF
   818  004120  004767  003164                  JSR     PC,     TYPE    ;GO TO TYPE "SET SWITHCHES
   819  004124  010350                  1$:     OPMODE                  ;FOR OPERATNG MODE IE.  PICKUPS
   820  004126  012767  010515  177770          MOV     #OPMOD2,1$
   821  004134  000000                          HALT                    ;OR DROPS OR BOTH THEN HIT CONTINUE"
   822  004136  032737  040000  177570          BIT     #BIT14,@#SWR    ;PARITY CHECKING BIT SET?
   823  004144  001402                          BEQ     BEGIN3          ;IF NOT BRANCH
   824  004146  004767  176874                  JSR     PC,HAMP         ;GO TO ENABLE PARITY
   825
   826                                          ;CONVERT THE TYPED LIMITS INTO HIGH AND LOW LIMITS
   827                                          ;IF SELF STARTING USED THEN "STBANK" AND "BANKS"
   828                                          ;HAVE ALREADY BEEN FILLED FOR USE HERE
   829
```

```
 838  004102  016700  176774          BEGIN3: MOV     STBANK, R0      ;MAKE WORKING LOCATION FOR LOW LIMIT
 839  004106  005200                          INC     R0
 840  004110  012703  177600                  MOV     #-200,  R3      ;R3 WILL HAVE PAGE ADDRESS
 841  004114  016701  177664                  MOV     BANKS,  R1      ;MAKE WORKING LOCATION FOR HIGH LIMIT
 842  004120  012702  160000                  MOV     #-20000,R2      ;R2 WILL HAVE LOW LIMIT
 843  004124  062702  020000          1$:     ADD     #20000, R2      ;GETTING LOW LIMIT
 844  004130  062703  000200                  ADD     #200,   R3      ;GETTING LOW LIMIT PAGE ADDRESS
 845  004134  077005                          SOB     R0,     1$      ;REPEAT R0 TIMES
 846  004136  010367  176516                  MOV     R3,     LOBASE  ;STORE LOW LIMIT PAGE ADDRESS
 847  004142  010267  176440                  MOV     R2,     LOLMT   ;SET LOW LIMIT
 848  004146  010267  176440                  MOV     R2,     MOD     ;SET MODULE NUMBER
 849  004152  162703  000200                  SUB     #200,   R3      ;GETTING READY FOR HIGH LIMIT PAGE
 850  004156  062702  020000          BEGIN4: ADD     #20000, R2      ;GETTING HIGH LIMIT
 851  004162  062703  000200                  ADD     #200,   R3      ;GETTING HIGH LIMIT PAGE ADDRESS
 852  004166  077105                          SOB     R1,     BEGIN4  ;REPEAT FOF BANK NUMBER
 853  004170  005302                          DEC     R2              ;GETTING HIGH LIMIT WITHIN BANK
 854  004172  010267  176412                  MOV     R2,     HILMT   ;STORE HIGH LIMIT
 855  004176  010367  176460                  MOV     R3,     HIBASE  ;STORE HIGH LIMIT PAGE ADDRESS
 856  004202  022767  000006  176672          CMP     #6,     STBANK  ;DOES STARTING BANK NEED SEGMENTATION?
 857  004210  103401                          BLO     1$              ;BRANCH IF YES
 858  004212  000407                          BR      2$
 859  004214  162767  000200  176436  1$:     SUB     #200,   LOBASE  ;SEGMENTATION REQUIREG SO
 860  004222  005167  176424                  COM     PASFLG          ;
 861  004226  000167  000716                  JMP     SEGMT1          ;SUBTRACT TO ENABLE REPEATED ADDITION
 862  004232  022703  001670          2$:     CMP     #1600,  R3      ;IS SEGMENTATION REQUIRED FOR HIGH?
 863  004236  101003                          BHI     BEGIN5          ;IF NO THEN BRANCH
 864  004240  012767  157776  176342          MOV     #157776,HILMT   ;IF YES HIGH LIMIT = END OF 32K
 865
 866                                  ;SETUP ALL TAGS TO THEIR APPROPIATE VALUES
 867
 868  004246  005067  176342          BEGIN5: CLR     COL             ;MAKE COLUMN ZERO
 869  004252  005067  176340                  CLR     ROW             ;MAKE ROW ZERO
 870  004256  005067  176336                  CLR     PCHRG           ;MAKE PRECHARGE ZERO
 871  004262  012767  023420  176332          MOV     #10000.,        CYCNT   ;SETUP CYCLE COUNT
 872  004270  012767  177777  176326          MOV     #-1,    TSTPAT  ;SETUP TEST PATTERN
 873  004276  012767  177777  176322          MOV     #-1,    BCKGRD  ;SETUP BACKGROUND
 874  004304  012767  000400  176320          MOV     #BIT08, MSKBIT  ;SETUP MASK
 875
 876
 877
 878  004312  032737  010000  177570  BEGIN6: BIT     #SW12,@#SWR     ;IS SWITCH 12 SET?
 879  004320  001413                          BNE     1$              ;IF SET BRANCH TO CHECK SW11
 880  004322  005167  176324                  COM     PASFLG          ;COMPLEMENT PASS FLAG
 881                                                                  ;IS BOTH PICKUPS AND DROPS DONE
 882  004326  001037                          BNE     3$              ;IF YES BRANCH
 883  004330  005067  176272                  CLR     BCKGRD          ;IF NOT SETUP TO
 884  004334  005067  176264                  CLR     TSTPAT          ;CATCH
 885  004340  012767  177777  176372          MOV     #-1,    OUTSID  ;BIT PICKS
 886  004346  000437                          BR      4$
 887  004350  005067  176276          1$:     CLR     PASFLG  ;
 888  004354  032737  010000  177570          BIT     #SW12,@#SWR     ;TEST SW12
 889  004362  001410                          BEQ     2$              ;IF NOT SET BRANCH TO CATCH DROPS
 890  004364  005067  176236                  CLR     BCKGRD          ;IF NOT SET TO CATCH PICKUPS
```



```
 891  004370  005067  176230                  CLR     TSTPAT          ;
 892  004374  012767  177777  176336          MOV     #-1,OUTSID      ;
 893  004402  000421                          BR      4$              ;
 894  004404  012767  177777  176212  2$:     MOV     #-1,    TSTPAT  ;
 895  004412  012767  177777  176206          MOV     #-1,    BCKGRD  ;
 896  004420  005067  176314                  CLR     OUTSID          ;
 897  004424  000416                          BR      4$
 898  004426  012767  177777  176170  3$:     MOV     #-1,    TSTPAT
 899  004434  012767  177777  176164          MOV     #-1,    BCKGRD
 900  004442  005067  176272                  CLR     OUTSID
 901  004446  005067  176056          4$:     CLR     SPASS           ;CLEAR PASS COUNT
 902
 903
 904
 905                                  ;MAKE THE TEST ADDRESS AND THE INSTRUCTION ADDRESS
 906
 907
 908
 909  004452  016746  176134          LOOP1:  MOV     MOD,    -(SP)   ;PICKUP MODULE NUMBER
 910  004456  042716  017777                  BIC     #17777,(SP)     ;STRIP AWAY ANY JUNK
 911  004462  016746  176126                  MOV     COL,-(SP)       ;PICKUP COLUMN NUMBER
 912  004466  042716  160777                  BIC     #^C17000,(SP)   ;STRIP AWAY JUNK
 913  004472  032767  020000  176114          BIT     #BIT13,COL      ;CHECK C4=1
 914  004500  001402                          BEQ     1$              ;BR IF NO
 915  004502  052716  000002                  BIS     #BIT01,(SP)     ;PUT C4 AT PROPER PLACE IN BUS ADDRESS
 916  004506  016746  176106          1$:     MOV     PCHRG,-(SP)     ;PICK UP PRECHARGE
 917  004512  042716  177763                  BIC     #^C14,(SP)      ;STRIP JUNK
 918  004516  016746  176074                  MOV     ROW,-(SP)       ;GET THE ROW NUMBER
 919  004522  042716  177017                  BIC     #^C760,(SP)     ;STRIP THE JUNK IF ANY
 920  004526  012608                          MOV     (SP)+,R0        ;FORM THE BUS ADDRESS WITH ROW
 921  004530  052608                          BIS     (SP)+,R0        ;PRECHARGE
 922  004532  052608                          BIS     (SP)+,R0        ;COLUMN
 923  004534  052600                          BIS     (SP)+,R0        ;MODULE
 924  004536  010067  176042                  MOV     R0,TSTADR       ;SAVE AS THE TEST ADDRESS
 925  004542  010067  176062                  MOV     R0,INSTAD       ;AND AS THE INSTRUCTION ADDRESS
 926
 927
 928
 929                                  ;MOVE EITHER  "0,-1,-1,"  OR THE  "CLEAR R0"
 930                                  ;INTO THE LOCATION INSTR1: THIS WILL FINALLY BE THE TEST INSTRUCTION
 931
 932
 933
 934  004546  010700          LPERR1: MOV     PC,     R0      ;THESE TWO INSTRUCTIONS
 935                                                          ;GETS THE LOCATION
 936  004550  062700  176154          ADD     #INSTR1-.,R0    ;OF INSTR1 IN TO R0
 937                                                          ;EVEN AFTER RELOCATION
 938  004554  013767  177570  000160          MOV     @#SWR,  SWITCH  ;GET SWITCH SETTING
 939                                                          ;TO CHECK FOR
 940  004562  042767  177577  000152          BIC     #177577,        SWITCH  ;IF SW07=0 DO OUTSIDE
 941
 942                                                          ;IF SW07=1 DO INSIDE
```

```
                                        BEQ     1$              ;IF SW07=0 THEN BRANCH

                                        MOV     PC,     R1      ;SW7=1 SO SETUP TO
                                                                ;MOVE INSIDE INTO INSTR1
                                        ADD     #INSIDE-.,R1    ;MAKE INSIDE POSITION
                                                                ;INDEPENDANT
                                        BR      2$

                                1$:     MOV     PC,     R1      ;SW07=0 SO SETUP TO
                                                                ;MOVE OUTSIDE INTO INSTR1
                                        ADD     #OUTSID-.,R1    ;MAKE OUTSIDE POSITION
                                                                ;INDEPENDANT
                                2$:     MOV     (R1)+,  (R0)+   ;THESE THREE INSTRUCTIONS

                                        MOV     (R1)+,  (R0)+   ;MOVES THE APPROPIATE

                                        MOV     (R1)+,  (R0)+   ;GROUP INTO INSTR1

                                        CLRB    SERFLG          ;CLEAR ERROR FLAG




                                ;FILL 4K WITH BACKGROUND PATTERN


                                FILL4K: MOV     BCKGRD, R0      ;GETTING READY TO MOVE
                                                                ;BACKGROUND INTO 4K
                                        MOV     MOD,    R1      ;STARTING FROM MODULE
                                                                ;NUMBER
                                        MOV     #10000, R2      ;10000=4K
                                1$:     MOV     R0,     (R1)+   ;FILL BACKGROUND
                                        SOB     R2,     1$      ;REPEAT 4K TIMES

                                        ;THIS MOVES CONTENTS OF TEST PATTERN (TSTPAT)
                                        ;HERE + =1 INTO MOD AND ALL LOCATIONS GOT
                                        ;BY INCREASING COLUMN BY ONE TILL ONE 4K
                                        ;IS COMPLETE
                                LDPAT:  MOV     @TSTADR,R0      ;MOVE TEST ADDRESS
                                        MOV     R0,R1           ;DUPLICATE IT IN R1 TO BE REINSTATED AFTER ADDING COLUMN
                                        BIC     #^C160774,R1    ;DON'T KEEP THE COL
                                        MOV     COL,R2          ;PICK UP THE COL
                                        MOV     #BIT13!BIT01,R3 ;GET READY TO TRANSFER BIT 13 TO BIT 1
                                        MOV     @TSTPAT,R4      ;MOVE TEST PATTERN=+1
                                1$:     ADD     #1000,R2        ;ADD ONE TO COLUMN
                                        BIC     #BIT14,R2       ;IF IT WAS ALREADY 37 THEN ADDING ONE WILL GIVE BIT 14
                                        MOV     R2,R5           ;FORM AN ADDRESS
                                        BIT     #BIT13,R5       ;POSITION C4 IF NEEDED
                                        BEQ     2$              ;IF BIT 13 IS ZERO THEN BRANCH
                                        XOR     R3,R5           ;IF BIT 13=1 THEN TRANSFER TO BIT 1
                                2$:     BIS     R1,R5           ;REINSTATE OTHER BITS AFTER ADDING TO COLUMN
```



```
                                        MOV     R4,(R5)         ;STORE THE PATTERN
                                        CMP     R0,R5           ;TSTADR?
                                        BNE     1$              ;BR IF NO


                                ;THE FOLLOWING JSR LOADS INSRTUCTION (INSTR1) INTO
                                ;PROPER PLACE


                                LOOP2:  JSR     PC,     LOINT

                                        BR      NEXT

                                        MOV     (R0),   SV400

                                        MOV     CYCNT,  R1      ;DO TEST SEQ. THIS
                                                                ;MANY TIMES
                                        TST     (PC)+
                                SWITCH: .WORD   0
                                        BEQ     1$
                                        MOV     INSTAD,-(SP)    ;SETUP THE STARTING ADDRESS
                                        MOV     (SP),@#DISPLAY  ;DISPLAY THE TEST ADDRESS
                                        JSR     PC,@(SP)+       ;GO TO THE TEST SEQ.
                                        BR      3$
                                1$:     MOV     INSTAD,R2
                                        MOV     R2,@#DISPLAY
                                2$:     MOV     (R2),(R0)
                                        SOB     R1,2$
                                3$:     MOV     SV400,(R0)


                                ;CHECK THE 1K BLOCK


                                        JSR     PC,CHECK        ;GO CHECK THE 1K BLOCK
                                        BR      NEXT            ;RETURN HERE ON ERROR
                                        BR      LOOP2
                                NEXT:   BIT     #BIT00,@#SWR    ;LOCK ON THIS PRECHARGE?
                                        BNE     1$              ;BR IF YES
                                        ADD     #4,PCHRG        ;GO TO NEXT PRECHARGE
                                        BIC     #^C14,PCHRG     ;GET RID OF TRASH
                                        BNE     LOOP1           ;LOOP IF MORE TO DO
                                1$:     BIT     #BIT01,@#SWR    ;LOCK ON THIS MODULE?
                                        BNE     SEOP            ;BR IF YES
                                        ADD     #BIT13,MOD      ;INCREMENT THE MOD
                                        BCS     SEGNT           ;IF ADDING BIT13 MAKES 0
                                                                ;THEN BRANCH
                                        CMP     MOD,HILMT       ;DON'T LET IT GET TO BIG
                                        BLO     LOOP12
                                        BR      SEGNT           ;GO TO CHECK IF SEGMENTATION IS REQUIRED
                                LOOP12: JMP     LOOP1           ;RETURN
                                ;IS SEGMENTATION REQUIRED?  IF SO TURN ON MEMORY
```

```
1046                                    ;MANAGEMENT AND SETUP ALL REQUIRED REGISTERS
1047                                    ;ONLY KERNEL MODE INSTRUCTION SPACE IS USED
1048
1049  005076  022767  001600  175556  SEGMT1: CMP     #1600.,  HIBASE  ;SEGMENTATION REQUIRED FOR HIGH LIMIT?
1050  005104  101114                          BHI     SEOP             ;IF NO THEN BRANCH
1051  005106  022767  001600  175544          CMP     #1600.,  LOBASE  ;IF YES THEN TEST IF LOW LIMIT REQ SEGMENTATION?
1052  005114  101415                          BLOS    SEGMT1           ;IF LOW LIMIT DOES REQ SEGMENTATION BRANCH
1053  005116  012767  001600  175534          MOV     #1600.,  LOBASE  ;IF NO MAKE LOW LIMIT END OF 32K
1054  005124  012767  140000  175454  SEG0:   MOV     #140000,LOLMT    ;LOLMT LOOKS AT PAGE PAGE 0R 6
1055  005132  012767  140000  175452          MOV     #140000,MOD      ;SAME WITH MODULE NUMBER
1056  005140  012767  157776  175442          MOV     #157776,HILMT    ;HIGH LIMIT LOOKS AT PAGE  END OF 4K
1057  005146  000420                          BR      SEGMT2           ;BRANCH TO ENABLE SEGMENTATION
1058  005150  062767  000200  175502  SEGMT1: ADD     #200.,   LOBASE  ;DO NEXT 4K
1059  005156  012767  140000  175422  SEG1:   MOV     #140000,LOLMT    ;MAKE LOW LIMIT AND MODULE
1060  005164  312767  140000  175420          MOV     #140000,MOD      ;0 ON PAGE "PAGE OR 6"
1061  005172  012767  157776  175410          MOV     #157776,HILMT    ;HIGH LIMIT SAME 4K ON PAGE
1062  005200  026767  175456  175452          CMP     HIBASE, LOBASE   ;IS IT THE END?
1063  005206  103453                          BLO     SEOP             ;IF YES BRANCH
1064  005210  016777  175444  175450  SEGMT2: MOV     LOBASE, @PAGE    ;SET LOW LIMIT ON PAGE 4
1065  005216  005077  172666                  CLR     @SR3             ;USE ONLY I SPACE IN KERNAL
1066  005222  012700  077406                  MOV     #77406, R0       ;MOVE INTO R0
1067  005226  010077  172660                  MOV     R0,      @KIPDR0 ;SEGMENT DESCRIPTOR
1068  005232  010077  172656                  MOV     R0,      @KIPDR1 ;REGISTER IS SET FOR
1069  005236  010077  172654                  MOV     R0,      @KIPDR2 ;128 BLOCKS WITH NO
1070  005242  010077  172652                  MOV     R0,      @KIPDR3 ;SYSTEM TRAPS,  THESE
1071  005246  010077  172650                  MOV     R0,      @KIPDR4 ;REGISTERS ARE ALSO
1072  005252  010077  172646                  MOV     R0,      @KIPDR5 ;CALLED PAGE
1073  005256  010077  172644                  MOV     R0,      @KIPDR6 ;DESCRIPTOR REGISTERS
1074  005262  010077  172642                  MOV     R0,      @KIPDR7 ;ONLY PAGE 0 AND 4 IS USED
1075  005266  005077  172640                  CLR     @KIPAR0 ;MAKE PAGE ADDRESS 0=0
1076  005272  012777  000001  172602          MOV     #1,      @SR0    ;ENABLE SEGMENTATION
1077  005300  032737  010000  177570          BIT     #SW12,   @#SWR   ;TEST SWITCH 12
1078  005306  001003                          BNE     2$               ;BRANCH IF SET
1079  005310  005167  175336                  COM     PASFLG           ;SW12 NOT SETTE, DO BOTH
1080  005314  000402                          BR      3$               ;SO COMPLEMENT AND RESTART
1081  005316  005067  175330          2$:     CLR     PASFLG           ;SW12 SET IE. DO ONE $O RESTART
1082  005322  000167  176764          3$:     JMP     BEGIN6
1083
1084
1085
1086                                    ;*****************************************************************
1087                                    ;END OF PASS ROUTINE
1088                                    ;INCREMENT THE PASS NUMBER
1089                                    ;IF SW10=0 PRINT END PASS ON END OF PROGRAM
1090                                    ;IF THERE IS A MONITOR GO TO IT,
1091                                    ;IF NONE JUMP TO BEGIN3
1092                                    ;IF LIMITS ARE NOT CHOSEN AND ALL OF MEMORY
1093                                    ;IS TO BE TESTED THIS RELOCATES PROGRAM FROM
1094                                    ;FIRST 4K TO SECOND 4K AND SETS UP FOR
1095                                    ;TESTING THE FIRST 4K,  THEN RELOCATES
1096                                    ;TO FIRST 4K TO BE ABLE TO START AGAIN.
1097
1098
1099
```



```
1100  005326  024042                  BEG2:   20000+BEGIN2
1101  005330  007736                  LENGTH: 7736
1102  005332  000100                  FROM:   FIRST
1103  005334  020100                  TO:     20000 + FIRST
1104
1105
1106
1107  005336  005767  175312          SEOP:   TST     SELF             ;WAS SELF STARTING USED
1108  005342  001523                          BEQ     SEOP1            ;IF NOT IE. LIMITS WERE CHOSEN THEN BRANCH
1109                                                                   ;WAS ENTRY INTO THIS SECTION BEEN MADE?
1110                                                                   ;IF YES IE. THIS IS SECOND TIME THEN BRANCH
1111  005344  005767  173752                  TST     SKT11            ;
1112  005350  100005                          BPL     5$               ;
1113  005352  005077  172924                  CLR     @SR0             ;
1114  005356  000240                          NOP
1115  005360  000240                          NOP
1116  005362  000240                          NOP
1117  005364  016700  177740          8$:     MOV     LENGTH, R0       ;IF NOT IE. FIRST TIME OR THIRD TIME
1118  005370  016701  177736                  MOV     FROM,   R1       ;GETTING READY TO RELOCATE
1119  005374  016702  177734                  MOV     TO,     R2       ;GETTING READY TO RELOCATE
1120  005400  012767  000000  175474          MOV     #0,     STBANK   ;FILL IN TYPED INFORMATION
1121  005406  012767  000001  175510          MOV     #1,     BANKS    ;ABOUT STARTING ADDRESSES
1122  005414  005067  175166                  CLR     LOLMT            ;SETUP LIMITS
1123  005420  005067  175166                  CLR     MOD              ;MODULE
1124  005424  012767  017776  175156          MOV     #17776, HILMT    ;AND HIGH
1125  005432  012122                  1$:     MOV     (R1)+,  (R2)+    ;FROM INTO TO
1126  005434  077002                          SOB     R0,     1$       ;REPEAT
1127  005436  005167  175214                  COM     REDONE           ;BEEN THROUGH ONCE?
1128  005442  016705  175210                  MOV     REDONE,R5        ;
1129  005446  001401                          BEQ     2$               ;BRANCH IF SECOND TIME THROUGH
1130  005450  000433                          BR      3$               ;FIRST TIME GO TO 3$
1131  005452  012737  000001  003102  2$:     MOV     #1,     @#STBANK
1132  005460  016737  175200  003124          MOV     LSTBK,  @#BANK0
1133  005466  012706  001100                  MOV     #1100,SP
1134  005472  000240                          NOP
1135  005474  010537  002656                  MOV     R5,@#REDONE
1136  005500  012737  000100  005332          MOV     #FIRST,@#FROM
1137  005506  012737  020100  005334          MOV     #20000+FIRST,@#TO
1138  005514  000240                          NOP
1139  005516  000240                          NOP
1140  005520  000240                          NOP
1141  005522  000240                          NOP
1142  005524  000240                          NOP
1143  005526  000240                          NOP
1144  005530  000137  005612                  JMP     @#SEOP1
1145  005534  000137  004102          4$:     JMP     @#BEGIN3
1146  005540                          3$:
1147  005540  012706  021100                  MOV     #21100,SP
1148  005544  010537  022656                  MOV     R5,@#20000+REDONE
1149  005550  012701  005332                  MOV     #FROM,R1         ;
1150  005554  012702  005334                  MOV     #TO,R2           ;
1151  005560  062701  020000                  ADD     #20000,R1        ;
1152  005564  062702  020000                  ADD     #20000,R2        ;
1153  005570  016703  177536                  MOV     FROM,R3          ;
```

```
1154  005574  016711  177534                    OV      TO,(R1)          ;
1155  005600  010312                            OV      R3,(R2)          ;
1156  005602  005137  022652                    COM     @#20000+PASFLG
1157  005606  000137  024312                    JMP     @#BEGIN6+20000
1158
1159  005612  005767  175034          $EOP1:    TST     PASFLG           ;
1160  005616  001011                            BNE     $EOP3            ;
1161  005620  005267  174714          $EOP2:    INC     SPASS            ;INCREMENT PASS COUNT
1162  005624  032737  002000  177570            BIT     #SW10,@#SWR      ;TYPE END PASS?
1163  005632  001003                            BNE     $EOP3            ;
1164  005634  004767  001400                    JSR     PC,TYPE
1165  005640  010335                            .PASS
1166  005642  005767  173454          $EOP3:    TST     $KT11
1167  005646  100002                            BPL     1$
1168  005650  005077  172226                    CLR     @SR0
1169  005654  000167  176222          1$:       JMP     BEGIN3
1170
1171
1172
1173
1174  005660                            LDINT:
1175  005660  010246                            MOV     R2,-(SP)                ;PUSH R2 ON STACK
1176  005662  010346                            MOV     R3,-(SP)                ;PUSH R3 ON STACK
1177  005664  016700  174740            1$:     MOV     INSTAD,R0               ;GET THE INSTRUCTION ADDRESS
1178  005670  042700  177017                    BIC     #^C760,R0               ;KEEP ROW
1179  005674  020067  174716                    CMP     R0,ROW                  ;AT THE TEST ROW?
1180  005700  001412                            BEQ     4$                      ;BR IF YES
1181  005702  016746  174722                    MOV     INSTAD,-(SP)
1182  005706  016776  174714  000000            MOV     BCKGRD,@(SP)            ;RESTORE BACK GROUND
1183  005714  062716  000002                    ADD     #2,(SP)
1184  005720  016736  174714                    MOV     SVIN2,@(SP)+
1185  005724  000403                            BR      5$
1186  005726  016777  174672  174674  4$:       MOV     ^TSTPAT,@INSTAD
1187  005734  016777  174664  174642  5$:       MOV     ^TSTPAT,@^TSTADR        ;STORE THE TEST PATTERN
1188  005742  016700  174662                    MOV     INSTAD,R0               ;GET THE INSTRUCTION8 ADDRESS
1189  005746  004767  000600            2$:     JSR     PC,FORMAD               ;GO FORM NEXT INSTUCTION ADDRESS
1190  005752  010003                            MOV     R0,R3                   ;CHECK FOR OUT OF RANGE
1191  005754  016702  174624                    MOV     ^TSTADR,R2              ;PICKUP THE ^TSTADR FOR CHECKING
1192  005760  020302                            CMP     R3,R2                   ;IS INSTR. ADDRESS = TEST ADDRESS?
1193  005762  001443                            BEQ     3$                      ;BR IF YES
1194  005764  062703  000002                    ADD     #2,R3                   ;ADDRESS WHERE INSTR2 GOES
1195  005770  020302                            CMP     R3,R2                   ;IS IT THE SAME AS TSTADR
1196  005772  001437                            BEQ     3$                      ;BR IF YES
1197  005774  062703  000002                    ADD     #2,R3                   ;INSTR3'S ADDRESS
1198  006000  020302                            CMP     R3,R2                   ;SAME AS TSTADR?
1199  006002  001433                            BEQ     3$                      ;BR IF YES
1200  006004  020367  174600                    CMP     R3,HILMT                ;WILL IT GO OFF THE END?
1201  006010  101356                            BHI     2$                      ;BR IF YES
1202  006012  010067  174612                    MOV     R0,INSTAD               ;SAVE NEW ADDRESS
1203  006016  011067  174614                    MOV     (R0),SVIN1              ;SAVE WHATS AT THE 1ST TWO
1204  006022  016067  000002  174610            MOV     2(R0),SVIN2             ;TEST INSTRUCTIONS ADDRESSES
1205  006030  016710  174670                    MOV     INSTR1,(R0)             ;LOAD THE INSTRUCTIONS
1206  006034  016760  174666  000002            MOV     INSTR2,2(R0)
1207  006042  016760  174662  000004            MOV     INSTR3,4(R0)
```



```
1208  006050  016746  174556                    MOV     MSKBIT,-(SP)            ;PICKUP THE MASK BIT
1209  006054  074016                            XOR     R0,(SP)                 ;HALF ADD TO THE ADDRESS
1210  006056  012600                            MOV     (SP)+,R0                ;GET THE ADDRESS TO BE WRITTEN INTO
1211  006060  020002                            CMP     R0,R2                   ;IS IT THE SAME AS THE TEST ADDRESS?
1212  006062  001700                            BEQ     1$                      ;BR IF YES
1213  006064  062766  000002  000004            ADD     #2,4(SP)                ;ADJUST THE RETURN ADDRESS
1214  006072                            3$:
1215  006072  012603                            MOV     (SP)+,R3                ;POP STACK INTO R3
1216  006074  012602                            MOV     (SP)+,R2                ;POP STACK INTO R2
1217  006076  000207                            RTS     PC                      ;GO BACK
1218
1219
1220
1221  006100  000000                    ^COL:   0
1222
1223  006102  010246                    CHECK:  MOV     R2,-(SP)                ;PUSH R2 ON STACK
1224  006104  010346                            MOV     R3,-(SP)                ;PUSH R3 ON STACK
1225  006106  010446                            MOV     R4,-(SP)                ;PUSH R4 ON STACK
1226  006110  010546                            MOV     R5,-(SP)                ;PUSH R5 ON STACK
1227  006112  016701  174466                    MOV     ^TSTADR,R1
1228  006116  016702  174504                    MOV     INSTAD,R2               ;ADDRESS OF THE TEST INSTR1
1229  006122  010200                            MOV     R2,R0
1230  006124  010203                            MOV     R2,R3
1231  006126  062703  000002                    ADD     #2,R3                   ;ADDRESS OF THE TEST INSTR2
1232  006132  010004                            MOV     R0,R4                   ;USED TO KEEP TRACK OF THE TEST ROW
1233  006134  042704  177017                    BIC     #^C760,R4
1234  006140  010067  177734                    MOV     R0,TCOL                 ;USED TO KEEP TRACK OF THE TEST COL
1235  006144  042767  160775  177726            BIC     #^C17000,TCOL
1236  006152  010746                            MOV     PC,     -(SP)           ;SETUP THE ESCAPE ON ERROR ADDRESS
1237  006154  062716  000352                    ADD     #CHK1-.,(SP)            ;MAKE IT POSITION INDEPENDANT
1238  006160  012667  174456                    MOV     (SP)+,ESCAPE
1239  006164  000407                            BR      1$
1240
1241  006166  042700  017762            8$:     BIC     #17762,R0               ;FORM ADDRESS
1242  006172  050400                            BIS     R4,R0                   ;GET THE ROW
1243  006174  056700  177700                    BIS     ^COL,R0                 ;AND THE COLUMN
1244  006200  020002                            CMP     R0,R2                   ;IS THIS THE INSTRUCTION ADDRESS?
1245  006202  001551                            BEQ     CHK1                    ;BR IF YES
1246
1247  006204  011005                    1$:     MOV     (R0),R5                 ;PICK UP THE DATA
1248  006206  020001                            CMP     R0,R1                   ;AT ^TSTADR?
1249  006210  001016                            BNE     2$                      ;BR IF NO
1250
1251  006212  020567  174406                    CMP     R5,TSTPAT               ;DID THE TSTPAT UNDER TEST CHANGE?
1252  006216  001515                            BEQ     7$                      ;BR IF NO
1253  006220  010567  174350                    MOV     R5,$BDDAT
1254  006224  016767  174374  174346            MOV     ^TSTPAT,$DDDAT
1255  006232  010067  174332                    MOV     R0,$BDADR
1256  006236  004767  000352                    JSR     PC,$HLT
1257  006242  000001                            1
1258  006244  000502                            BR      7$
1259
1260  006246  020002                    2$:     CMP     R0,R2                   ;AT INSTAD
1261  006250  001020                            BNE     4$                      ;BR IF NO
```

```
1262  006252  020567  174446                  CMP     R5,INSTR1               ;DID INSTR1 UNDER TEST CHANGE?
1263  006256  001412                          BEQ     3$                      ;BR IF NO
1264  006260  010567  174310                  MOV     R5,$BDDAT
1265  006264  016767  174434  174300          MOV     INSTR1,$BDDAT
1266  006272  010067  174272                  MOV     R0,$BDADR
1267  006276  004767  000312                  JSR     PC,$HLT
1268  006302  000002                          2
1269  006304  016710  174326          3$:     MOV     $VIN1,(R0)              ;RESTORE THIS LOCATION
1270  006310  000460                          BR      7$
1271
1272  006312  020003                  4$:     CMP     R0,R3                   ;AT THE 2ND INSTRUCTION
1273  006314  001020                          BNE     6$                      ;BR IF NO
1274  006316  020567  174404                  CMP     R5,INSTR2               ;DID IT CHANGE?
1275  006322  001412                          BEQ     5$                      ;BR IF NO
1276  006324  010567  174244                  MOV     R5,$BDDAT
1277  006330  016767  174372  174234          MOV     INSTR2,$GDDAT
1278  006336  010067  174226                  MOV     R0,$BDADR
1279  006342  004767  000246                  JSR     PC,$HLT
1280  006346  000002                          2
1281  006350  016710  174264          5$:     MOV     $VIN2,(R0)              ;RESTORE THIS INSTRUCTION
1282  006354  000436                          BR      7$
1283  006356  026704  174234          6$:     CMP     ROW,R4                  ;ON THE SAME ROW AS TSTADR?
1284  006362  001016                          BNE     10$
1285  006364  026705  174234                  CMP     TSTPAT,R5               ;DID THE PATTERN CHANGE?
1286  006370  001430                          BEQ     7$
1287  006372  010567  174176                  MOV     R5,$BDDAT
1288  006376  016767  174222  174166          MOV     TSTPAT,$GDDAT
1289  006404  010067  174160                  MOV     R0,$BDADR
1290  006410  004767  000200                  JSR     PC,$HLT
1291  006414  000001                          1
1292  006416  000415                          BR      7$
1293  006420  020567  174202          10$:    CMP     R5,BCKGRD               ;DID BACK GROUND CHANGE
1294  006424  001412                          BEQ     7$                      ;BR IF NO
1295  006426  010567  174142                  MOV     R5,$BDDAT
1296  006432  016767  174170  174132          MOV     BCKGRD,$GDDAT
1297  006440  010067  174124                  MOV     R0,$BDADR
1298  006444  004767  000144                  JSR     PC,$HLT
1299  006450  000003                          3
1300
1301  006452  062704  000020          7$:     ADD     #20,R4                  ;GO TO THE NEXT ROW
1302  006456  042704  177017                  BIC     #^C760,R4               ;DON'T LET IT GET TO BIG
1303  006462  001241                          BNE     8$                      ;GO DO NEXT ROW IF NOT TIME FOR A NEW COL
1304
1305  006464  062767  001000  177456  11$:    ADD     #1000,TCOL              ;GO TO THE NEXT COL
1306  006472  042767  140775  177400          BIC     #^C37002,TCOL           ;DON'T LET IT GET TO BIG
1307  006500  032767  020000  177372          BIT     #BIT13,TCOL
1308  006506  001627                          BEQ     8$                      ;IF MORE TO DO GO DO IT
1309  006510  010146                          MOV     R1,-(SP)
1310  006512  012701  020002                  MOV     #BIT13!BIT01,R1
1311  006516  074167  177356                  XOR     R1,TCOL
1312  006522  012601                          MOV     (SP)+,R1
1313  006524  000620                          BR      8$
1314
1315  006526  012605          CHK1:   MOV     (SP)+,R5                ;POP STACK INTO R5
```



```
1316  006530  012604                          MOV     (SP)+,R4                ;POP STACK INTO R4
1317  006532  012603                          MOV     (SP)+,R3                ;POP STACK INTO R3
1318  006534  012602                          MOV     (SP)+,R2                ;POP STACK INTO R2
1319  006536  105767  174612                  TSTB    SERFLG                  ;WAS THERE ANY ERRORS?
1320  006542  001002                          BNE     EXIT                    ;IF YES EXIT
1321  006544  062716  000002                  ADD     #2,(SP)                 ;SET FOR NO ERRORS EXIT
1322  006550  000207          EXIT:   RTS     PC
1323
1324
1325
1326
1327  006552  010046          FORMAD: MOV     R0,-(SP)                ;SAVE R0
1328  006554  042700  160015                  BIC     #^C17762,R0             ;STRIP EVERYTHING BUT THE ROW AND COL
1329  006560  062700  000020                  ADD     #20,R0                  ;ADD A ROW
1330  006564  032700  020000                  BIT     #BIT13,R0               ;C4
1331  006570  001405                          BEQ     1$                      ;BR IF NO
1332  006572  010146                          MOV     R1,-(SP)
1333  006574  012701  020002                  MOV     #BIT13!BIT01,R1
1334  006600  074100                          XOR     R1,R0
1335  006602  012601                          MOV     (SP)+,R1
1336  006604  042716  017762          1$:     BIC     #17762,(SP)             ;STRIP THE ROW AND COL
1337  006610  052600                          BIS     (SP)+,R0                ;FORM ADDRESS
1338  006612  000207                          RTS     PC
1339
1340
1341                                  ;*****************************************************************
1342                                  ;THIS ROUTINE IS THE HLT HANDLER
1343                                  ;SW10=1 BELL ON ERROR
1344                                  ;SW13=1 INHIBIT ERROR TYPEOUTS
1345                                  ;SW15=1 HALT ON ERROR
1346                                  ;GO TO TYPERR ON ERROR
1347  006614                  $HLT:
1348  006614  032737  002000  177570          BIT     #SW10,@#SWR             ;BELL ON ERROR?
1349  006622  001403                          BEQ     1$                      ;NO = SKIP
1350  006624  004767  000410                  JSR     PC,TYPE
1351  006630  002644                          $BELL
1352  006632  005267  173714          1$:     INC     SERTTL                  ;COUNT THE NUMBER OF ERRORS
1353  006636  001775                          BEQ     1$                      ;INSURE SERTTL NOT=0
1354  006640  032737  020000  177570          BIT     #SW13,@#SWR             ;SKIP TYPEOUT IF SET
1355  006646  001015                          BNE     2$                      ;SKIP TYPEOUTS
1356  006650  011667  173710                  MOV     (SP),$HLTAD             ;GET ADDRESS OF HLT INSTRUCTION
1357  006654  117767  173704  173700          MOVB    @$HLTAD,$ITEMB          ;SAVE THE ERROR ITEM CODE
1358  006662  162767  000002  173674          SUB     #2,$HLTAD
1359  006670  004767  000046                  JSR     PC,TYPERR               ;GO TO USER ERROR ROUTINE
1360  006674  004767  000340                  JSR     PC,TYPE
1361  006700  002647                          SCRLF
1362  006702  005737  177570          2$:     TST     @#SWR                   ;HALT ON ERROR
1363  006706  100001                          BPL     3$                      ;SKIP IF CONTINUE
1364  006710  000000                          HALT                            ;HALT ON ERROR!
1365  006712  112767  177777  173634  3$:     MOVB    #-1,SERFLG
1366  006720  062716  000002                  ADD     #2,(SP)
1367  006724  032737  001000  177570          BIT     #BIT09,@#SWR
1368  006732  001002                          BNE     4$
1369  006734  016716  173702                  MOV     ESCAPE,(SP)
```

```
1370  006740  000207                  4$:     RTS     PC
1371
1372  006742  010046                  TYPERR: MOV     R0,-(SP)                ;PUSH R0 ON STACK
1373  006744  010146                          MOV     R1,-(SP)                ;PUSH R1 ON STACK
1374  006746  116700  173610                  MOVB    $ITEMB,R0               ;PICKUP THE ITEM NUMBER
1375  006752  005300                          DEC     R0                      ;FORM AN INDEX FOR THE TABLE
1376  006754  006300                          ASL     R0
1377  006756  006300                          ASL     R0
1378  006760  006300                          ASL     R0
1379  006762  010746                          MOV     PC,-(SP)
1380  006764  062716  173710                  ADD     #$ERRTB-.,(SP)
1381  006770  062600                          ADD     (SP)+,R0
1382  006772  012067  000012                  MOV     (R0)+,1$                ;PICKUP THE ERROR MESSAGE
1383  006776  004767  000236                  JSR     PC,TYPE
1384  007002  002647                          $CRLF
1385  007004  004767  000230                  JSR     PC,TYPE
1386
1387  007010  000000                  1$:     0                               ;ERROR MESSAGE POINTER GOES HERE
1388  007012  012067  000014                  MOV     (R0)+,2$                ;PICKUP THE DATA HEADER POINTER
1389  007016  032737  000400  177570          BIT     #BIT08,@#SWR            ;TYPE THE DATA HEADER?
1390  007024  001003                          BNE     11$                     ;BR IF NO
1391  007026  004767  000206                  JSR     PC,TYPE
1392  007032  000000                  2$:     0
1393  007034  012001                  11$:    MOV     (R0)+,R1                ;GET THE DATA STRING POINTER
1394  007036  011000                          MOV     (R0),R0                 ;GET THE DATA FORMAT POINTER
1395  007040  010746                          MOV     PC,     -(SP)
1396  007042  062716  170736                  ADD     #-.,    (SP)
1397  007046  061600                          ADD     (SP),   R0
1398  007050  062601                          ADD     (SP)+,  R1
1399  007052  016767  173552  173516          MOV     INSTAD,$TMP0            ;SETUP THE COLUMN FOR TYPING
1400  007060  000367  173512                  SWAB    $TMP0
1401  007064  006267  173506                  ASR     $TMP0
1402  007070  042767  177740  173500          BIC     #^C37,$TMP0
1403  007076  016767  173526  173474          MOV     INSTAD,$TMP1            ;SET THE ROW FOR TYPING
1404  007104  006267  173470                  ASR     $TMP1
1405  007110  006267  173464                  ASR     $TMP1
1406  007114  006267  173460                  ASR     $TMP1
1407  007120  006267  173454                  ASR     $TMP1
1408  007124  042767  177740  173446          BIC     #^C37,$TMP1
1409  007132  016767  173472  173442          MOV     INSTAD,$TMP2            ;SETUP PRECHARGE FOR TYPING
1410  007140  006267  173436                  ASR     $TMP2
1411  007144  006267  173432                  ASR     $TMP2
1412  007150  042767  177774  173424          BIC     #^C3,$TMP2
1413  007156  000410                          BR      9$
1414  007160  005711                  10$:    TST     (R1)                    ;TERMINATOR?
1415  007162  001003                          BNE     3$                      ;BR IF NO
1416  007164  012601                          MOV     (SP)+,R1                ;POP STACK INTO R1
1417  007166  012600                          MOV     (SP)+,R0                ;POP STACK INTO R0
1418  007170  000207                          RTS     PC
1419  007172  004067  000074          3$:     JSR     R0,$SPACE               ;GO TYPE SOME SPACES
1420  007176  000000                  4$:     .WORD   0                       ;HOW MANY GOES HERE
1421  007200  112067  000030          5$:     MOVB    (R0)+,6$                ;PICKUP NUMBER OF DIGITS TO TYPE
1422  007204  112067  000025                  MOVB    (R0)+,7$                ;PICKUP THE LEADING ZEROS FLAG
1423  007210  112067  177762                  MOVB    (R0)+,4$                ;PICKUP THE NUMBER OF SPACES
```



```
1424  007214  012146                          MOV     (R1)+,-(SP)             ;PICKUP THE DATA
1425  007216  010746                          MOV     PC,-(SP)
1426  007220  062716  170560                  ADD     #-.,(SP)
1427  007224  062616                          ADD     (SP)+,(SP)
1428  007226  013646                          MOV     @(SP)+,-(SP)
1429  007230  004067  000064                  JSR     R0,$B20CT               ;GO TYPE AN OCTAL DIGIT
1430  007234     000          6$:     .BYTE   0                               ;HOW MANY DIGITS
1431  007235     000          7$:     .BYTE   0                               ;LEADING ZERO SWITCH
1432  007236  000750                          BR      1$                      ;LOOP
1433
1434                                  ;*****************************************************************
1435
1436                                  ;THIS IS A SMALL ROUTINE USED TO MAKE THE TYPE POINTERS POSITION INDEPENT
1437
1438  007240  010746                  TYPE:   MOV     PC,-(SP)                ;GET THE POINTER
1439  007242  062716  170536                  ADD     #-.,(SP)                ;GET THE PC
1440  007246  067616  000002                  ADD     @2(SP),(SP)             ;STRIP THE RELATIVE ADDRESS
1441  007252  012667  000004                  MOV     (SP)+,1$                ;AND SAVE IT FOR TYPE OUT
1442  007256  004767  171626                  JSR     PC,$TYPE                ;GO TYPE IT
1443  007262  000000                  1$:     .WORD   0                       ;POINTER TO MESSAGE
1444  007264  062716  000002                  ADD     #2,(SP)                 ;SETUP TO RETURN
1445  007270  000207                          RTS     PC                      ;GO BACK
1446
1447                                  ;*****************************************************************
1448                                  ;ROUTINE TO TYPE SPACES ON THE TTY
1449                                  ;CALL:
1450                                  ;       JSR     R0,$SPACE
1451                                  ;       NUMBER OF SPACES
1452                                  ;       RETURN HERE
1453                                  ;
1454  007272  012067  000002          $SPACE: MOV     (R0)+,2$                ;GET NUMBER OF SPACES DESIRED
1455                                                                          ;AND SET RETURN ADR.
1456  007276  005327                  1$:     DEC     (PC)+                   ;FINISHED?
1457  007300  000000                  2$:     0
1458  007302  002001                          BGE     3$                      ;BR IF NO
1459  007304  000200                          RTS     R0                      ;GO HOME
1460  007306                          3$:
1461  007306  004767  177726                  JSR     PC,TYPE
1462  007312  007316                          4$
1463  007314  000770                          BR      1$
1464  007316     040          000     4$:     .BYTE   40,0                    ;STORAGE FOR ONE SPACE AND TERMINATOR
1465                                  ;*****************************************************************
1466                                  ;BINARY TO OCTAL (ASCII) AND TYPE
1467                                  ;$B20CT---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
1468                                  ;CALL:
1469                                  ;       MOV     NUM,-(SP)               ;NUMBER TO BE TYPED
1470                                  ;       JSR     R0,$B20CT               ;CALL FOR TYPEOUT
1471                                  ;       .BYTE   N                       ;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
1472                                  ;       .BYTE   M                       ;M=0 TO 3
1473                                                                          ;       0=SUPPRESS LEADING ZEROS
1474                                                                          ;       1=TYPE LEADING ZEROS
1475                                                                          ;       2=SUPPRESS LEADING ZEROS
1476                                                                          ;       WITH PHYSICAL ADDRESS
1477                                                                          ;       3=TYPE LEADING ZEROS
```

```
                                                                                ;WITH PHYSICAL ADDRESS
                                        ;
                                        ;$B201=---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST $BSOCT OR $B2016
                                        ;CALL:
                                        ;       MOV     NUM,-(SP)
                                        ;       JSR     R0,$B201
                                        ;
                                        ;$B2016---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
                                        ;CALL:
                                        ;       MOV     NUM,-(SP)
                                        ;       JSR     R0,$B2016
                                        ;
007320  112067  000267          $B2OCT: MOVB    (R0)+,$OMODE+1          ;PICKUP THE NUMBER OF DIGITS TO TYPE
007324  112067  000261                  MOVB    (R0)+,$0FILL            ;GET THE ZERO FILL SWITCH
007330  000406                          BR      $B201
007332  112767  000001  000251  $B2016: MOVB    #1,$0FILL               ;SET THE ZERO FILL SWITCH
007340  112767  000006  000245          MOVB    #6,$OMODE+1             ;SET FOR SIX(6) DIGITS
007346  112767  000005  000234  $B201:  MOVB    #5,$OCNT                ;SET THE ITERATION COUNT
007354  010346                          MOV     R3,-(SP)                ;SAVE R3
007356  010446                          MOV     R4,-(SP)                ;SAVE R4
007360  010546                          MOV     R5,-(SP)                ;SAVE R5
007362  116704  000225                  MOVB    $OMODE+1,R4             ;GET THE NUMBER OF DIGITS TO TYPE
007366  005404                          NEG     R4
007370  062704  000006                  ADD     #6,R4                   ;SUBTRACT IT FOR MAX. ALLOWED
007374  110467  000212                  MOVB    R4,$OMODE               ;SAVE IT FOR USE
007400  116704  000205                  MOVB    $0FILL,R4               ;GET THE ZERO FILL SWITCH
007404  042704  177776                  BIC     #177776,R4              ;
007410  016605  000010                  MOV     10(SP),R5               ;PICKUP THE INPUT NUMBER
007414  005003                          CLR     R3                      ;CLEAR THE OUTPUT WORD
007416  132767  000002  000165          BITB    #2,$0FILL               ;
007424  001424                          BEQ     1$                      ;
007426  005767  171670                  TST     $KT11
007432  100021                          BPL     1$
007434  032777  000001  170440          BIT     #1,@#RB                 ;
007442  001415                          BEQ     1$                      ;
007444  017746  173216                  MOV     @PAGE,-(SP)             ;
007450  042705  160000                  BIC     #160000,R5              ;
007454  006316                          ASL     (SP)                    ;
007456  006316                          ASL     (SP)                    ;
007460  006316                          ASL     (SP)                    ;
007462  006316                          ASL     (SP)                    ;
007464  006316                          ASL     (SP)                    ;
007466  006103                          ROL     R3                      ;
007470  006316                          ASL     (SP)                    ;
007472  006103                          ROL     R3                      ;
007474  062605                          ADD     (SP)+,R5                ;
007476  006105                  1$:     ROL     R5                      ;ROTATE MSB INTO "C"
007500  000404                          BR      3$                      ;GO DO MSB
007502  006105                  2$:     ROL     R5                      ;FORM THIS DIGIT
007504  006105                          ROL     R5
007506  006105                          ROL     R5
007510  010503                          MOV     R5,R3
007512  006103                  3$:     ROL     R3                      ;GET LSB OF THIS DIGIT
007514  105367  000072                  DECB    $OMODE                  ;TYPE THIS DIGIT?
```



```
007520  100017                          BPL     7$                      ;BR IF NO
007522  042703  177770                  BIC     #177770,R3              ;GET RID OF JUNK
007526  001002                          BNE     4$                      ;TEST FOR 0
007530  005704                          TST     R4                      ;SUPPRESS THIS 0?
007532  001403                          BEQ     5$                      ;BR IF YES
007534  005204                  4$:     INC     R4                      ;DON'T SUPPRESS ANYMORE 0'S
007536  052703  000060                  BIS     #'0,R3                  ;MAKE THIS DIGIT ASCII
007542  052703  000040          5$:     BIS     #' ,R3                  ;MAKE ASCII IF NOT ALREADY
007546  110367  000034                  MOVB    R3,8$                   ;SAVE FOR TYPING
007552  004767  177462                  JSR     PC,TYPE
007556  007606                          8$
007560  105367  000024          7$:     DECB    $OCNT                   ;COUNT BY 1
007564  003346                          BGT     2$                      ;BR IF MORE TO DO
007566  002402                          BLT     6$                      ;BR IF DONE
007570  005204                          INC     R4                      ;INSURE LAST DIGIT ISN'T A BLANK
007572  000743                          BR      2$                      ;GO DO THE LAST DIGIT
007574  012605                  6$:     MOV     (SP)+,R5                ;RESTORE R5
007576  012604                          MOV     (SP)+,R4                ;RESTORE R4
007600  012603                          MOV     (SP)+,R3                ;RESTORE R3
007602  012616                          MOV     (SP)+,(SP)              ;SET THE STACK FOR RETURNING
007604  000200                          RTS     R0                      ;RETURN
007606     000          8$:     .BYTE   0                               ;STORAGE FOR ASCII DIGIT
007607     000                  .BYTE   0                               ;TERMINATOR FOR TYPE ROUTINE
007610     000          $OCNT:  .BYTE   0                               ;OCTAL DIGIT COUNTER
007611     000          $0FILL: .BYTE   0                               ;ZERO FILL SWITCH
007612  000000          $OMODE: 0                                       ;NUMBER OF DIGITS TO TYPE
```

```
1558
1559
1560                                    ;*****************************************************************************
1561                                    ;*****************************************************************************
1562                                    ;ASCII MESSAGES
1563  007614  042524  052123  052040  EM1:   .ASCIZ  /TEST PATTERN/<15><12>
1564  007622  052101  042524  047122
1565  007630  005015     000
1566  007633     111  051516  051124  EM2:   .ASCIZ  /INSTRUCTION CHANGED/<15><12>
1567  007640  041525  044524  047117
1568  007646  041440  040510  043516
1569  007654  042105  005015     000
1570  007661     102  041501  020113  EM3:   .ASCIZ  /BACK GROUND CHANGED/<15><12>
1571  007666  051107  052517  042116
1572  007674  041440  040510  043516
1573  007702  042105  005015     000
1574  007707     105  051122  051117  DH1:   .ASCII  /ERROR   FAILING/<15><12>
1575  007714  020040  043040  044501
1576  007722  044514  043516  005015
1577  007730  041520  020040  020040          .ASCIZ  /PC      ADDRESS EXPECT  RCV'D   COL  ROW  PRECHARGE/<15><12>
1578  007736  020040  042101  051104
1579  007744  051505  020123  054105
1580  007752  042520  052103  020040
1581  007760  041522  023526  020104
1582  007766  020040  047503  020114
1583  007774  051040  053517  020040
1584  010002  051120  041505  040510
1585  010010  043522  006505  000012
1586  010016  051105  047522  020122  DH2:   .ASCII  /ERROR   INST    EXPECT  RCV'D/<15><12>
1587  010024  020040  047111  052123
1588  010032  020040  020040  054105
1589  010040  042520  052103  020040
1590  010046  041522  023526  006504
1591  010054     012
1592  010055     120  020103  020040          .ASCIZ  /PC      ADDRESS DATA    DATA    COL  ROW  PRECHARGE/<15><12>
1593  010062  020040  040440  042104
1594  010070  042522  051523  042040
1595  010076  052101  020101  020040
1596  010104  042040  052101  020101
1597  010112  020040  041440  046117
1598  010120  020040  047522  020127
1599  010126  050040  042522  044103
1600  010134  051101  042507  005015
1601  010142     000
1602  010143     105  051122  051117  DH3:   .ASCII  /ERROR   BCKGRD  EXPECT  RCV'D/<15><12>
1603  010150  020040  041040  045503
1604  010156  051107  020104  042440
1605  010164  050130  041505  020124
1606  010172  051040  053103  042047
1607  010200  005015
1608  010202  041520  020040  020040          .ASCIZ  /PC      ADDRESS DATA    DATA    COL  ROW  PRECHARGE/<15><12>
1609  010210  020040  042101  051104
1610  010216  051505  020123  040504
1611  010224  040524  020040  020040
```



```
1612  010232  040504  040524  020040
1613  010240  020040  047503  020114
1614  010246  051040  053517  020040
1615  010254  051120  041505  040510
1616  010262  043522  006505  000012
1617                          ;EVEN
1618  010270  002564  002570  002572  DT1:   .WORD   SHLTAD,SBDADR,SGDDAT,SBDDAT,STMP0,STMP1,STMP2,0
1619  010276  002574  002576  002600
1620  010304  002602  000000
1621  010310     006     001     002  DF1:   .BYTE   6,1,2
1622  010313     006     003     002          .BYTE   6,3,2
1623  010316     006     001     002          .BYTE   6,1,2
1624  010321     006     001     003          .BYTE   6,1,3
1625  010324     002     000     003          .BYTE   2,0,3
1626  010327     002     000     005          .BYTE   2,0,5
1627  010332     002     000     000          .BYTE   2,0,0
1628  010335     105  042116  050040  EPASS:  .ASCIZ  /END PASS/<15><12>
1629  010342  051501  006523  000012
1630  010350  042523  020124  053523  OPMODE: .ASCII  /SET SWITCHES FOR OPERATING MODE REFER SECTION 4.1/<15><12>
1631  010356  052111  044103  051505
1632  010364  043040  051117  047440
1633  010372  042520  040522  044524
1634  010400  043516  046440  042117
1635  010406  020105  042522  042506
1636  010414  020122  042523  052103
1637  010422  047511  020116  027064
1638  010430  006461     012
1639  010433     040  042511  020056          .ASCIZ  / IE. PICKUPS OR DROPS OR BOTH THEN HIT CONTINUE/<15><12>
1640  010440  044520  045503  050125
1641  010446  020123  051117  042040
1642  010454  047522  051520  047440
1643  010462  020122  047502  044124
1644  010470  052040  042510  020116
1645  010476  044510  020124  047503
1646  010504  052116  047111  042525
1647  010512  005015     000
1648  010515     123  052105  051440  OPMOD2: .ASCIZ  /SET SWITCHES/<15><12>
1649  010522  044527  041524  042510
1650  010530  006523  000012
1651  010534  054524  042520  021040  STARTB: .ASCII  /TYPE "STARTING BANK" NUMBER AND CR /<15><12>
1652  010542  052123  051101  044524
1653  010550  043516  041040  047101
1654  010556  021113  047040  046525
1655  010564  042502  020122  047101
1656  010572  020104  051103  006440
1657  010600     012
1658  010601     122  043105  051105          .ASCIZ  /REFER TO SECTION 4.3.1 BEFORE CHOOSING /<15><12>
1659  010606  052040  020117  042523
1660  010614  052103  047511  020116
1661  010622  027064  027063  020061
1662  010630  042502  047506  042522
1663  010636  041440  047516  051517
1664  010644  047111  020107  005015
1665  010652     000
```

```
1666  010653    000123  040524  052122  START2: .ASCIZ  /STARTING BANK?/<15><12>
1667  010660    047111  020107  040502
1668  010666    045516  006477  000012
1669  010674    054524  042520  047040  NUMBER: .ASCIZ  /TYPE NUMBER OF 4K BANKS TO BE TESTED AND CR /<15><12>
1670  010702    046525  042502  020122
1671  010710    043117  032040  020113
1672  010716    040502  045516  020123
1673  010724    047524  041040  020105
1674  010732    042524  052123  042105
1675  010740    040440  042116  041440
1676  010746    020122  005015  000
1677  010753    000043  047440  020106  NUMB2:  .ASCIZ  /# OF BANKS?/<15><12>
1678  010760    040502  045516  037523
1679  010766    005015  000
1680  010771    000114  051501  020124  MEMEND: .ASCIZ  /LAST MEMORY ADDRESS IS /
1681  010776    042515  047515  054522
1682  011004    040440  042104  042522
1683  011012    051523  044440  020123
1684  011020    000
1685  011021    000120  051101  052111  PAERR:  .ASCIZ  /PARITY ERROR DETECTED AT LOC.  /
1686  011026    020131  051105  047522
1687  011034    020122  042504  042524
1688  011042    052103  042105  040440
1689  011050    020124  047514  027103
1690  011056    020040  000
1691  011061    000111  020116  040520  NOPARE: .ASCIZ  /IN PARITY TEST NO PARITY ERROR DECTED/<15><12>
1692  011066    044522  054524  052040
1693  011074    051505  020124  047516
1694  011102    050040  051101  052111
1695  011110    020131  051105  047522
1696  011116    020122  042504  052103
1697  011124    042105  005015  000
1698  011131    000120  051101  052111  PARERR: .ASCIZ  /PARITY ERROR HAS OCCURED SO PARITY TEST HAS STARTED/<15><12>
1699  011136    020131  051105  047522
1700  011144    020122  040510  020123
1701  011152    041517  052503  042522
1702  011160    020104  047523  050040
1703  011166    051101  052111  020131
1704  011174    042524  052123  044040
1705  011202    051501  051440  040524
1706  011210    052122  042105  005015
1707  011216    000
1708  011217    000123  051117  054522  SORRY:  .ASCIZ  /SORRY LAST INPUT INVALID TRY AGAIN/<15><12>
1709  011224    046040  051501  020124
1710  011232    047111  052520  020124
1711  011240    047111  040526  044514
1712  011246    020104  051124  020131
1713  011254    043501  044501  006516
1714  011262    000012
1715  011264    051120  043517  040522  PROTO:  .ASCII  /PROGRAM RELOCATED/<15><12>
1716  011272    020115  042522  047514
1717  011300    040503  042524  006504
1718  011306    000012
1719  011307    000122  051505  040524          .ASCIZ  /RESTART ADDRESS   /
```



```
1720  011314  052122  040440  042104
1721  011322  042522  051523  020040
1722  011330  000040
1723  011332  047524  050040  052125  PROBAK: .ASCIZ  /TO PUT PROGRAM BACK TO FIRST 4K START  /
1724  011340  050040  047522  051107
1725  011346  046501  041040  041501
1726  011354  020113  047524  043040
1727  011362  051111  052123  052040
1728  011370  020113  052123  051101
1729  011376  020124  000040
1730  011402  047524  051040  051505  RELOD:  .ASCIZ  /TO RESTORE LOADER START  /
1731  011410  047524  042522  046040
1732  011416  040517  042504  020122
1733  011424  052123  051101  020124
1734  011432  000040
1735  011434  050040  052514  020123  RELOD2: .ASCIZ  / PLUS RELOCATION FACTOR/<15><12>
1736  011442  042522  047514  040503
1737  011450  044524  047117  043040
1738  011456  041501  047524  006522
1739  011464  000012
1740  011466  047514  042101  051105  FIN:    .ASCIZ  /LOADER RESTORED/<15><12>
1741  011474  051040  051505  047524
1742  011502  042522  006504  000012
1743                                  .EVEN
1744  011510  000000                  LAST:   .WORD   0
1745          000001                  .END
```

```
BANKS   003124          BCKGRD  002626          BEGIN1  002746          BEGIN2  004042
BEGIN3  004102          BEGIN4  004156          BEGIN5  004246          BEGIN6  004312
BEGN11  003454          BEG2    005326          BIT0    = 000001        BIT00   = 000001
BIT01   = 000002        BIT02   = 000004        BIT03   = 000010        BIT04   = 000020
BIT05   = 000040        BIT06   = 000100        BIT07   = 000200        BIT08   = 000400
BIT09   = 001000        BIT1    = 000002        BIT10   = 002000        BIT11   = 004000
BIT12   = 010000        BIT13   = 020000        BIT14   = 040000        BIT15   = 100000
BIT2    = 000004        BIT3    = 000010        BIT4    = 000020        BIT5    = 000040
BIT6    = 000100        BIT7    = 000200        BIT8    = 000400        BIT9    = 001000
BPTVEC  = 000014        CHECK   036102          CHK1    006526          COL     002614
CYCNT   002622          DF1     010310          DH1     007707          DH2     010016
DH3     010143          DISPLA  = 177570        DT1     010270          EMTVEC  = 000030
EM1     007614          EM2     007633          EM3     007661          EPASS   010335
ERRVEC  = 000004        ESCAPE  002642          EXIT    006550          FILL4K  004622
FIN     011466          FIRST   000100          FORMAD  006552          FROM    005332
FRSTAD  000216          HIBASE  002662          HILMT   002610          INSIDE  002732
INSTAD  002630          INSTR1  002724          INSTR2  002726          INSTR3  002730
IN2AD   002214          IOTVEC  = 000020        KIPAR0  000132          KIPAR1  000134
KIPAR2  000136          KIPAR3  000140          KIPAR4  000142          KIPAR5  000144
KIPAR6  000146          KIPAR7  000150          KIPDR0  000112          KIPDR1  000114
KIPDR2  000116          KIPDR3  000120          KIPDR4  000122          KIPDR5  000124
KIPDR6  000126          KIPDR7  000130          LAST    011510          LDINT   005468
LDPAT   004642          LENGTH  005330          LOADER  000324          LOBASE  002668
LODER   002672          LOLMT   002606          LOOP1   004452          LOOP12  005072
LOOP2   004722          LPERR   024546          LSTBK   002664          MAMF    002176
MEMEND  010771          MMVEC   = 000250        MOD     002612          MSKBIT  002632
NEXT    005212          NOPARE  011061          NUMBER  010674          NUMB2   010753
OPMODE  010350          OPMOD2  010515          OUTSID  002740          PAERR   011021
PAGE    002666          PARCSR  = 172100        PARERR  011131          PARSRV  002254
PARVEC  = 000114        PASFLG  002652          PC      =%000007        PCHRG   002628
PROBAK  011332          PROTO   011264          PS      = 177776        PSW     = 177776
PWRVEC  = 000024        REDONE  002656          RELOD   011402          RELOD2  011434
RESTOR  000300          RESVEC  = 000010        ROW     002616          R0      =%000000
R1      =%000001        R2      =%000002        R3      =%000003        R4      =%000004
R5      =%000005        R6      =%000006        R7      =%000007        SEGMT   005076
SEGMT1  005150          SEGMT2  005210          SEGVEC  000152          SEG0    005124
SEG10   005156          SELF    002654          SORRY   011217          SP      =%000006
SR0     000102          SR1     000104          SR2     000106          SR3     000110
START8  010534          START2  010653          STBANK  003102          SVIN1   002636
SVIN2   002640          SV400   002634          SWITCH  004742          SWR     = 177570
SW0     = 000001        SW00    = 000001        SW01    = 000002        SW02    = 000004
SW03    = 000010        SW04    = 000020        SW05    = 000040        SW06    = 000100
SW07    = 000200        SW08    = 000400        SW09    = 001000        SW1     = 000002
SW10    = 002000        SW11    = 004000        SW12    = 010000        SW13    = 020000
SW14    = 040000        SW15    = 100000        SW2     = 000004        SW3     = 000010
SW4     = 000020        SW5     = 000040        SW6     = 000100        SW7     = 000200
SW8     = 000400        SW9     = 001000        TBITVE  = 000014        TCOL    = 006100
TKB     001772          TKS     001770          TO      005334          TO1     002676
TRAPVE  = 000034        TRTVEC  = 000014        TSTADR  002684          TSTPAT  002624
TYPE    007240          TYPERR  006742          $BDADR  002578          $BODAT  002574
$BELL   002644          $B20CT  007320          $B201   007346          $B2016  007332
$CORE   001652          $COROU  001706          $CRLF   002647          $EOP    005336
$EOP1   005612          $EOP2   005620          $EOP3   005642          $ERFLG  002554
$ERRTB  002674          $ERTTL  002552          $FILLS  001105          $GDADR  002566

TEST MOS MEMORY MACY11.624 1-JUN-73 11:33 PAGE 42
DCMSB,P11 SYMBOL TABLE

$GDDAT  002572          $HLT    006614          $HLTAD  002564          $ICNT   002544
$ITEMB  002562          $KTOUT  001646          $KT11   001322          $LF     002658
$LPADR  002546          $LPERR  002550          $LSTAD  001764          $LSTBK  001766
$NULL   001104          $OCNT   007610          $OMODE  007612          $PASS   002548
$QUES   002646          $SEG    001326          $SIZE   001300          $SIZEX  001712
$SPACE  007272          $TMP0   002576          $TMP1   002600          $TMP2   002602
$TPB    001102          $TPFLG  001106          $TPS    001100          $TSTNM  002542
$TYPE   001110          $0FILL  007611          .RECD   001774          .       = 011512


 ERRORS DETECTED: 0

TEST MOS MEMORY MACY11.624 1-JUN-73 11:33 PAGE 43
DCMSB,P11

 *DCMSB,DCMSB/SOL=DCMSB
 RUN-TIME: 5 10 0 SECONDS
 CORE USED: 4K
```