MRV11-BA

LSI-11-UVPROM-RAM TEST

MD-11-DVMRA-A

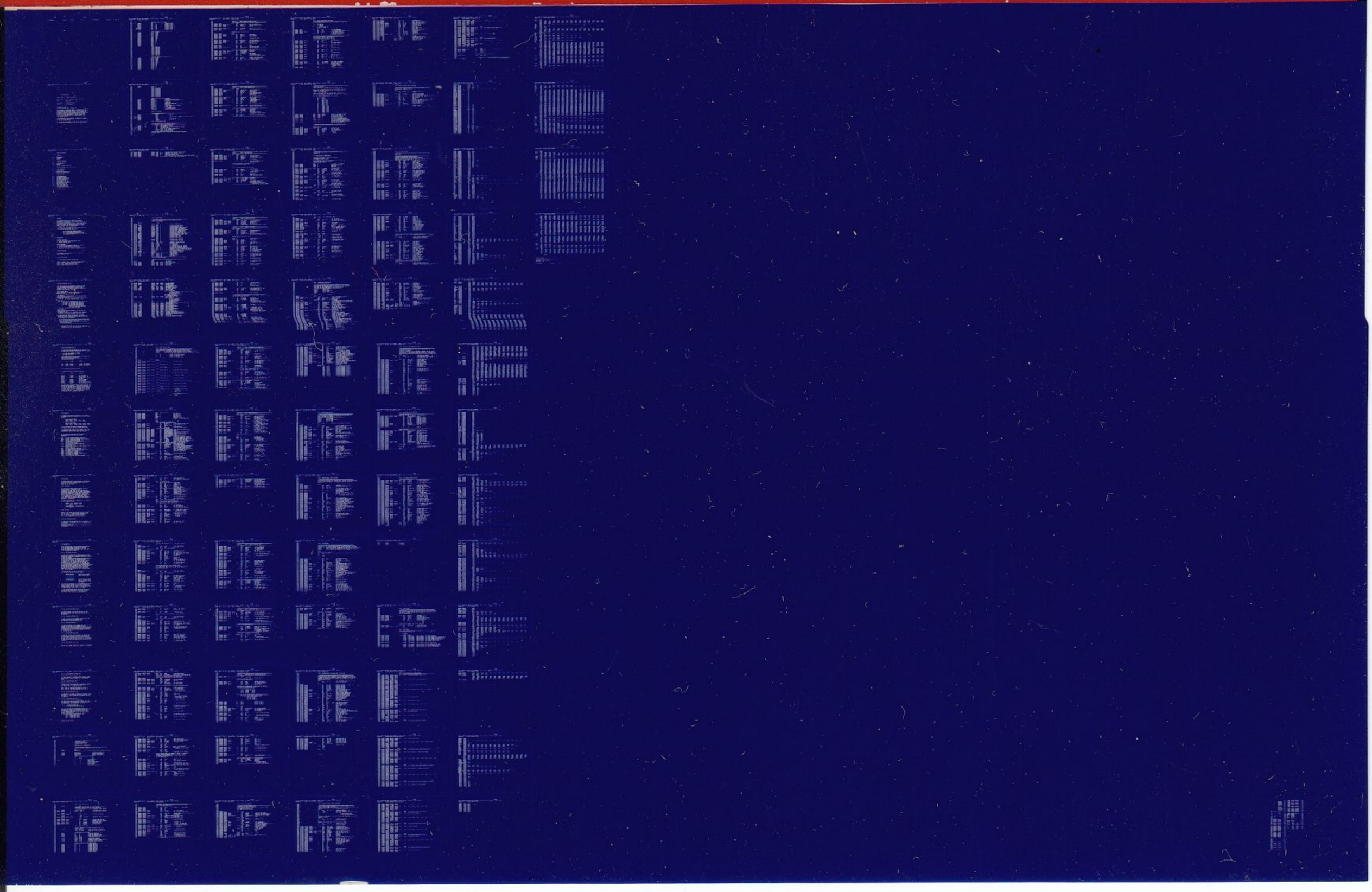
EP-DVMRA-A-DL-A

FICHE 1 OF 1

JUN 1977

digital

MADE IN USA



B01

EOF1DGM9AASE0	00010000	770526	PDP10 411	HDR1DVMRAASE0	00010000	770526

```
.REM %
```

IDENTIFICATION

PRODUCT CODE: MAINDEC-11-DVMRA-A-D

PRODUCT NAME: LSI11 UV PROM-RAM (MRV11-BA) TEST.

DATE: APRIL 1977

MAINTAINER: DIAGNOSTIC GROUP

AUTHOR: GUS PASQUANTONIO

TABLE OF CONTENTS

1.0 ABSTRACT

THIS PROGRAM CONSISTS OF A SERIES OF ROUTINES
DESIGNED TO TEST THE MRV11-BA UV PROM-RAM OPTION ON
AN LSI11 OR 11/03 SYSTEM.
THE PROGRAM ALSO PROVIDES OPTIONAL MEMORY DUMP ROUTINES
WHICH WILL OUTPUT THE CONTENTS OF ANY GIVEN MEMORY
RANGE, IN EITHER OCTAL OR BINARY FORMAT, OR OUTPUT A
32K MEMORY MAP. SEE 7.0 FOR DETAILS.

THE PROGRAM WILL RUN UNDER ACT, APT, AND/OR XXDP WITH
THE FOLLOWING RESTRICTIONS (SEE 4.3):

1. THE 256 WORD RAM ADDRESS JUMPERS MUST BE SET TO THE DEFAULT ADDRESS 020000.
2. THE 4K PROM ADDRESS JUMPERS MUST BE SET TO THE DEFAULT ADDRESS 140000, AND THE PROM SIZE JUMPERS SET FOR 4K.

2.0 REQUIREMENTS

2.1 EQUIPMENT REQUIRED:

1. LSI11 OR 11/03 WITH 4K READ/WRITE MEMORY.
2. CONSOLE TERMINAL (LA36, OR TTY).
3. MRV11-BA UV PROM-RAM.

2.2 MEMORY REQUIRED:

1. 4K READ/WRITE FOR PROGRAM STORAGE AND EXECUTION.
2. 256 WORD RAM (STANDARD IN MRV11-BA).
3. 1K TO 4K UV PROM (OPTIONAL IN MRV11-BA).
4. SPECIAL 4K TEST PROM (IN-HOUSE ENGINEERING USE ONLY).

3.0 LOADING PROCEDURE

USE STANDARD LOAD PROCEDURE FOR PDP-11 ABSOLUTE
BINARY FORMATTED TAPE.

4.0 STARTING PROCEDURE

FIVE START-RESTART ADDRESS ARE PROVIDED:

00200 = INITIAL START. INITIALIZE PROM-RAM ADDRESSES.
00204 = RESTART. RE-USE OLD PROM-RAM ADDRESSES.
00210 = OCTAL MEMORY DUMP START (SEE 7.1).
00214 = BINARY MEMORY DUMP START (SEE 7.1).
00220 = MEMORY MAP DUMP START (SEE 7.2).

STARTING PROCEDURE (CONT'D)

4.1 UPON INITIAL START THE PROGRAM WILL IDENTIFY ITSELF,
TYPE THE CURRENT SWITCH REGISTER CONTENTS (SEE 5.0),
AND BEGIN A BRIEF CONVERSATION IN WHICH THE RAM
ADDRESS, PROM SIZE, AND PROM ADDRESS ARE ESTABLISHED.
RESPOND TO EACH QUERY AS FOLLOWS:

"SWR = 000000 NEW = "
ENTER THE DESIRED SWITCH REGISTER CONTENTS AND
TERMINATE WITH <CR>. SEE 5.0 FOR DETAILS.

"RAM ADDRESS = "
ENTER THE OCTAL STARTING ADDRESS OF THE 256 WORD RAM,
AND TERMINATE WITH <CR>.
ADDRESS MUST BE ON A 256 WORD (1000 OCTAL BYTE) BOUNDARY.
IF <CR> ONLY, THE DEFAULT ADDRESS 020000 IS USED.

"PROM SIZE = "
ENTER THE PROM SIZE AND TERMINATE WITH <CR>, AS FOLLOWS:

```
        1 (OR 1024) = 1K OPTIONAL PROM INSTALLED.
        2 (OR 2048) = 2K OPTIONAL PROM INSTALLED.
        3 (OR 3072) = 3K OPTIONAL PROM INSTALLED.
        4 (OR 4096) = 4K OPTIONAL PROM INSTALLED.
        X           = SPECIAL 4K TEST PROM INSTALLED.
                      (IN-HOUSE ENGINEERING USE ONLY)
```

IF <CR> ONLY, THE DEFAULT SIZE 4K IS USED.

"PROM ADDRESS = "
ENTER THE OCTAL STARTING ADDRESS OF THE 4K PROM, AND
TERMINATE WITH <CR>.
THE ADDRESS MUST BE ON A 1K (4000 OCTAL BYTE) BOUNDARY.
IF <CR> ONLY, THE DEFAULT ADDRESS 140000 IS USED.

ALL INPUT DATA ARE TESTED FOR VALIDITY (ON PROPER
BOUNDARY AND NOT NON-EXISTANT ADDRESS) BEFORE PROCEEDING
TO THE NEXT QUERY. WHEN AN INVALID INPUT IS MADE, THAT
INPUT REQUEST WILL BE REPEATED.

****
NOTE: VALID PROM ENTRIES MUST BE MADE EVEN THOUGH THERE
MAY BE NO PROM PHYSICALLY INSTALLED.
THE PROM TESTS (11 AND 12) WILL RUN WITH OR WITHOUT
PROM CHIPS INSTALLED.
****

4.2 SUBSEQUENT RE-STARTS FROM LOC 204 WILL OMIT THE INITIAL
DIALOGUE AND USE THE SIZE AND ADDRESS PARAMETERS
OBTAINED AT THE LAST INITIALIZATION.

STARTING PROCEDURE (CONT'D)

4.3 THE FOLLOWING PROM-RAM PARAMETERS ARE ASSUMED BY THE PROGRAM UNLESS OTHERWISE ESTABLISHED AS IN 4.1 ABOVE:

1. THE FIRST RAM ADDRESS IS 020000.
2. THE FIRST PROM ADDRESS IS 140000.
3. THE PROM SIZE IS 4K.

THESE DEFAULT VALUES MAY BE ALTERED IF NECESSARY TO MEET SOME SPECIFIC CONFIGURATION.
CHANGE THE FOLLOWING MEMORY LOCATIONS TO CONFORM TO YOUR PARTICULAR SYSTEM:

```
LOC     TAG     CONTENT         COMMENT
---     ---     -------         -------

1250    $BASE:  020000          ;DEFAULT RAM ADDRESS.
1254    $CDW1:  140000          ;DEFAULT PROM ADDRESS.
1256    $CDW2:  000004          ;DEFAULT PROM SIZE (K).
```

5.0 SWITCH REGISTER OPTIONS AND CONTROL.

THE FOLLOWING SWITCH OPTIONS ARE PROVIDED:

SWITCH	OCTAL	FUNCTION
SWR15=1	100000	HALT ON ERROR.
SWR14=1	040000	LOOP ON TEST.
SWR13=1	020000	INHIBIT ERROR TYPEOUTS
SWR12=1		NOT USED.
SWR11=1	004000	INHIBIT ITERATIONS.
SWR10=1	002000	BELL ON ERROR.
SWR09=1	001000	LOOP ON ERROR.
SWR08=1	000400	LOOP ON TEST SWR <7:0>
SWR<7:0>	0004XX	TEST # FOR SWR 8 OPTION.

THE LSI11 USES A DEDICATED MEMORY LOCATION (00176) AS A SOFTWARE SWITCH REGISTER. THIS LOCATION FUNCTIONS THE SAME AS THE HARDWARE SWITCH REGISTER IN OTHER PDP-11'S. THE SOFT SWITCH REGISTER IS DISPLAYED AND/OR CHANGED AT INITIAL START TIME, AND THEREAFTER MAY BE ALTERED BY TYPING "CNTRL G" <↑G>. THE PROGRAM WILL TYPE THE CURRENT SWR CONTENTS. ENTER THE NEW VALUE, IF ANY, AND TERMINATE WITH "CARRIAGE RETURN" <CR>.

6.0 ERROR REPORTING

ANY ERRORS ENCOUNTERED ARE REPORTED ON THE CONSOLE TTY.
EACH ERROR SIGNATURE IS CONSISTENT WITH THE FOLLOWING
EXAMPLES;

```
RAM READ-WRITE ERROR.
TEST#   ERRPC   ADDR    GOOD    BAD
000DMN  XXXXXX  YYYYYY  177777  000000

PROM-RAM COPY ERROR.
TEST#   ERRPC   PRMADD  PRMDAT  RAMADD  RAMDAT
000DMN  XXXXXX  140000  010000  023000  000000
```

TESTING CONTINUES FOLLOWING ANY ERROR UNLESS THE
LOOP (SWR 9) OR HALT (SWR 15) OPTIONS ARE SELECTED.

IF AN UNFORSEEN BUS TIME-OUT TRAP OCCURS AT ANY TIME,
AN APPROPRIATE ERROR MESSAGE WILL BE OUTPUT, AND THE
PROCESSER WILL HALT AT LOC "STOP" (SEE LISTING).

THE FOLLOWING KEY WORDS ARE USED TO IDENTIFY THE
VARIOUS DATA WORDS THAT MAKE UP AN ERROR
SIGNATURE:

ADDR	THE WORD ADDRESS CURRENTLY UNDER TEST.
ADDR-2	THE NEXT LOWER WORD ADDRESS (TESTS 5 AND 6).
ADDR+2	THE NEXT HIGHER WORD ADDRESS (TESTS 5 AND 6).
BAD	THE BAD (FAILING) DATA WORD.
BUFDAT	THE CONTENTS OF AN INTERNAL BUFFER COPY OF RAM LOCATION "ADDR" (TEST 4).
BYTE	THE FAILING BYTE ADDRESS (TEST 10).
DATIS	THE CONTENTS OF PROM LOC "ADDR" AFTER A WRITE ATTEMPT (TEST 12).
DATWAS	THE CONTENTS OF PROM LOC "ADDR" BEFORE A WRITE ATTEMPT (TEST 12).
ERRPC	THE PC AT WHICH THE ERROR CALL OCCURRED.
GOOD	THE GOOD (EXPECTED) DATA WORD.
PATRN	THE CURRENT TEST BIT PATTERN (TESTS 5 AND 6).
PRMADD	THE CURRENT PROM ADDRESS (TEST 11).
PRMDAT	THE CONTENTS OF "PRMADD".
RAMADD	THE CURRENT RAM ADDRESS (TEST 11).
RAMDAT	THE CONTENTS OF "RAMADD" (TESTS 4 AND 11).
TEST#	THE CURRENT TEST NUMBER.
TRAPPC	THE PC AT WHICH A BUS-ERROR OCCURRED.
TRAPPS	THE PROCESSOR STATUS WHEN BUS-ERROR OCCURRED.

7.0 MISCELLANEOUS

TWO MEMORY DUMP ROUTINES ARE INCLUDED IN THE PROM-RAM TEST FOR CONVENIENCE. THESE ROUTINES WILL DUMP THE CONTENTS OF ANY GIVEN MEMORY RANGE, OR DUMP A 32K MEMORY MAP ON THE CONSOLE TTY.

7.1 MEMORY DATA DUMP.

THIS ROUTINE IS STARTED FROM LOC 00210 (FOR OCTAL OUTPUT) OR 00214 (FOR BINARY OUTPUT).
THE OPERATOR WILL BE ASKED TO INPUT THE STARTING ADDRESS AND THE WORD COUNT OF THE RANGE TO BE DUMPED.
ODD ADDRESS INPUTS ARE ACCEPTED, BUT THE DUMP RANGE WILL START AT THAT ADDRESS -1. WORD COUNT INPUT MUST BE GREATER THAN ZERO. THE DUMP RANGE IS CHECKED FOR VALIDITY, AND IF OK, THAT RANGE WILL BE TYPED ON THE TTY, 4 WORDS/LINE OCTAL, OR 2 WORDS/LINE BINARY.
IF THE DUMP RANGE IS INVALID (NON-EXISTANT ADDRESS OR WORD COUNT = 0) THE STARTING ADDRESS AND WORD COUNT QUERIES WILL BE REPEATED.

ON OUTPUT, EACH DATA SET IS PRECEEDED BY THE ADDRESS OF THE FIRST WORD OF THE SET. I.E.

```
 0      000000  111111  022222  133333
10      044444  155555  066666  177777
              OR
 0      0000000000000000  1001001001001001
 4      0010010010010010  0011011011011011
              ETC.
```

7.2 MEMORY MAP DUMP.

NORMALLY A 32K MEMORY MAP IS OUTPUT AS AN INTEGRAL PART OF TEST #1. FOR MAINTENANCE PURPOSES, A START FROM LOC 00220 WILL FORCE A SINGLE PASS ENTRY TO TEST #1, OUTPUT THE RESULTING MAP, AND HALT.
SEE 9.1 FOR A DESCRIPTION OF THE MAP FORMAT.

8.0 EXECUTION TIME AND END PASS

THE FIRST PASS THRU THE PROGRAM IS A "QUICK VERIFY" WITH NO ITERATIONS. THEREAFTER, EACH TEST IS REPEATED 100 OCTAL TIMES.
EXECUTION TIME WILL VARY FROM 70 TO 150 SECONDS PER PASS DEPENDING ON PROM SIZE.
THE PROGRAM WILL TYPE "END PASS # X" AT THE CONCLUSION OF EACH PASS.

9.0 TEST DESCRIPTION

AFTER INITIALIZATION, THE TEST SEQUENCE IS CONDUCTED IN THE FOLLOWING ORDER. EACH TEST IS REPEATED 100 OCTAL TIMES BEFORE PROCEEDING TO THE NEXT. ANY ERRORS OR DISCREPANCIES ARE REPORTED ON THE CONSOLE TTY AS DESCRIBED IN 6.0 ABOVE.

9.1 TEST 1. 32K MEMORY MAP TEST.

FOR THIS TEST A MEMORY MAP IS CONSTRUCTED BY TESTING MEMORY FROM 0 THRU 32K, IN 256 WORD INCREMENTS. TESTING IS DONE VIA A R[illegible]D/WRITE SEQUENCE ON THE FIRST WORD OF OF EACH 256 SE[illegible]NT.
THE MAP THUS O[illegible]AINED DESCRIBES EACH 256 WORD SEGMENT AS NON-EXISTANT, READ-ONLY, OR READ/WRITE MEMORY.
THE MAP INVOLVES THE USE OF 2 WORDS FOR EACH 4K BANK (16. WORDS). ON THE INITIAL PASS EACH MAP PAIR IS TRANSLATED INTO A 16 CHARACTER STRING CO[illegible]OSED OF THE CHARACTERS (R) FOR READ ONLY, (W) FOR READ/WRITE, AND (-) FOR NON-EXISTANT, AND OUTPUT AS SHOWN BELOW.
ON SUBSEQUENT PA[illegible]ES, A N[illegible] MAP IS CONSTRUCTED AND COMPARED AGAINST THE ORIGINAL (REFE[illegible]NCE) MAP. ANY DIFFERENCES ARE INTERPRETED AS ERR[illegible]RS AND REPORTED ACCORDINGLY.
NORMALLY THE MAP IS PRINTED ONLY ONCE, UPON INITIAL START FROM LOC 000[illegible]0, AND NOT AT ALL IF RUNNING UNDER ACT11, APT11, [illegible]R XXDP.

A TYPICAL MAP FOR A 4K LSI-11, WITH MRV11-BA, AND 4K PROM MIGHT APPEAR AS FOLLOWS (COMMENTS ADDED):

```
WWWWWWWWWWWWWWWW        BANK 0 IS R/W MEMORY.
W---------------        RAM AT DEFAULT 020000
----------------        BANKS 2-5 ARE NEXM
----------------
----------------
----------------
RRRRRRRRRRRRRRRR        PROM AT DEFAULT 140000
-----R-----R----        BANK 7 (I/O PAGE) WITH
                        BOOTSTRAP ROM AT 165000,
                        AND 173000.
```

NOTE THAT EACH CHARACTER IN A LINE REPRESENTS A 256 WORD (1000 OCTAL BYTE) SEGMENT OF THAT BANK, STARTING AT ADDRESS X+00000 (LEFTMOST CHAR), AND ENDING AT ADDRESS X+17000 (RIGHTMOST CHAR), WHERE X = THE BANK BITS FOR THAT BANK (00, 02, 04, 06, 10, 12, 14, OR 16).

IF A MAP COMPARE ERROR OCCURS, THE MAP PAIR FOR THE FAILING BANK ARE TRANSLATED AS ABOVE AND THE RESULTING MAP LINE IS OUTPUT ON THE TTY. ONE SHOULD ALWAYS SAVE A COPY OF THE 1ST PASS MAP FOR REFERENCE.

9.2 TEST 2. RAM ONES AND ZEROS TEST.

THIS TEST VERIFIES THAT THE RAM CAN HOLD ALL ONES,
AND THEN ALL ZEROS. THE RAM IS FILLED WITH THE
APPROPRIATE DATA PATTERN, THEN EACH LOCATION IS READ
AND VERIFIED.

9.3 TEST 3. RAM BASIC ADDRESS TEST.

THE RAM IS FILLED WITH AN ADDRESS PATTERN WHERE EACH
LOCATION CONTAINS ITS OWN ADDRESS WORD.
EACH ADDRESS IS THEN READ AND VERIFIED.

9.4 TEST 4. RAM FAST READ TEST.

THIS TEST IS SIMILAR TO THE ADDRESS TEST ABOVE, IN THAT
THE DATA PATTERN USED IS AN ADDRESS PATTERN.
HOWEVER, IN THIS CASE THE ENTIRE RAM IS COPIED INTO AN
INTERNAL BUFFER USING 256 CONSECUTIVE "MOV" INSTRUCTIONS.
THE INTENT HERE IS TO HIT THE RAM AS FAST AS POSSIBLE.
THE CONTENTS OF THE RAM ARE THEN COMPARED TO THAT OF
THE INTERNAL BUFFER.

9.5 TEST 5. RAM SLIDING 0 BIT TEST.

IN THIS TEST THE RAM IS FILLED WITH 1'S, THEN A WORD
CONTAINING A SINGLE 0 BIT IS WRITTEN, BEGINNING WITH
BIT 0 OF WORD 0, AND ENDING WITH BIT 15 OF WORD 255.
AS EACH 0 BIT IS WRITTEN, THAT LOCATION (N) IS VERIFIED,
AND THE TWO ADJACENT LOCATIONS (N-2 AND N+2) ARE TESTED
TO VERIFY THAT THEY WERE UNAFFECTED BY THE WRITING INTO
LOCATION N.

9.6 TEST 6. RAM SLIDING 1 BIT TEST.

SAME AS TEST 4 ABOVE, EXCEPT THAT THE RAM IS INITIALIZED
WITH 0'S, AND A WORD CONTAINING A SINGLE 1 BIT IS WRITTEN.

9.7 TEST 7. RAM ALTERNATE BIT/WORD TEST.

THE RAM IS FILLED WITH 052525, 125252, 052525, 125252, ETC., AND VERIFIED. THE PATTERN IS THEN COMPLIMENTED 125252, 052525, 125252, 052525, ETC., AND VERIFIED AGAIN.

9.8 TEST 10. RAM WRITE BYTE TEST.

THE RAM IS FILLED WITH 1'S, AND A LO BYTE 0 IS WRITTEN INTO EACH WORD. EACH ADDRESS IS THEN CHECKED TO SEE THAT THE HI BYTE = -1, AND THE LO BYTE = 0.

REFILL WITH 1'S, AND WRITE AND CHECK A HI BYTE 0.
REFILL WITH 0'S, AND WRITE AND CHECK A LO BYTE -1.
REFILL WITH 0'S, AND WRITE AND CHECK A HI BYTE -1.

9.9 TEST 11. PROM TO RAM COPY TEST.

THE CONTENTS OF THE PROM IS COPIED INTO THE RAM, IN 256 WORD PACKETS. THEN THE RAM DATA IS COMPARED WITH THE PROM DATA. THE COPY-COMPARE CONTINUES UNTIL THE PROM IS EXHAUSTED.

9.10 TEST 12. PROM WRITE-TRAP TEST.

THIS TEST CHECKS THAT ANY ATTEMPT TO WRITE INTO THE PROM RESULTS IN A "BUS ERROR TRAP". ALL PROM ADDRESSES AS DETERMINED BY THE "PROM SIZE", ARE TESTED.

9.11 TEST 13. PROM DATA PATTERN TEST
(IN-HOUSE ENGINEERING USE ONLY)

TEST 13 IS INCLUDED FOR ENGINEERING PURPOSES ONLY, AND REQUIRES THAT A SPECIAL 4K PROGRAMMED PROM CHIP SET BE INSTALLED. ENTRY INTO THE TEST IS ONLY MADE IF THE "Y" RESPONSE WAS GIVEN TO THE "PROM SIZE" QUERY DURING INITIALIZATION.
THE TEST READS EACH PROM LOCATION AND VERIFIES THAT THE FOLLOWING BINARY COUNT PATTERN IS READ:

1ST K	010000 THRU 011777
2ND K	022000 THRU 023777
3RD K	044000 THRU 045777
4TH K	106000 THRU 107777

10.0 PROGRAM LISTING FOLLOWS:

```
                        %
   487
   488                                  .TITLE  LSI-11 UVPROM-RAM TEST. MD-11-DVMRA-A.
   489                                  ;*COPYRIGHT (C) 1977
   490                                  ;*DIGITAL EQUIPMENT CORP.
   491                                  ;*MAYNARD, MASS. 01754
   492                                  ;*
   493                                  ;*PROGRAM BY GUS PASQUANTONIO
   494                                  ;*
   495                                  ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
   496                                  ;*PACKAGE (MAINDEC-11-DZQAC-C2), SEPT 14, 1976.
   497                                  ;*
   498          000001                  $TN=1
   499          160000                  $SWR=160000     ;;HALT ON ERROR, LOOP ON TEST, INHIBIT ERROR TYPOUT
   500
   501          167400                  $SWR=167400                     ;REDEFINE SWITCHES USED.
   502          020000                  ABASE=020000                    ;DEFAULT RAM ADDRESS
   503          140000                  ACDW1=140000                    ;DEFAULT PROM ADDRESS
   504          000004                  ACDW2=4                         ;DEFAULT PROM SIZE IN K.
   505                                  .SBTTL  OPERATIONAL SWITCH SETTINGS
   506                                  ;*
   507                                  ;*      SWITCH                  USE
   508                                  ;*      ------          ------------------
   509                                  ;*        15            HALT ON ERROR
   510                                  ;*        14            LOOP ON TEST
   511                                  ;*        13            INHIBIT ERROR TYPEOUTS
   512                                  ;*        11            INHIBIT ITERATIONS
   513                                  ;*        10            BELL ON ERROR
   514                                  ;*         9            LOOP ON ERROR
   515                                  ;*         8            LOOP ON TEST IN SWR<7:0>
```

```
                                .SBTTL  TRAP CATCHER

                        000000          .=0
                                ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
                                ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
                                ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
                        000174          .=174
000174  000000         DISPREG: .WORD  0               ;;SOFTWARE DISPLAY REGISTER
000176  000000         SWREG:   .WORD  0               ;;SOFTWARE SWITCH REGISTER

                        000004          .=4
000004  006676  000000          .WORD   TIMOUT,0        ;BUS ERROR PROC'  ER.

                        000100          .=100
000100  000104  000200  000002  .WORD   104,200,2      ;IGNORE ANY "BEVNT" INTERRUPTS

                        000200          .=200
000200  000137  001436          JMP     @#START         ;INITIAL START AND INIT.
000204  000137  002620          JMP     @#RSTRT         ;RESTART. USE SAME PROM/RAM
                                                        ;PARAMETERS. NO MAP OUTPUT.
000210  000137  007174          JMP     @#ODUMP         ;OCTAL DUMP START
000214  000137  007276          JMP     @#BDUMP         ;BINARY DUMP START
000220  000137  007124          JMP     @#MDUMP         ;MAP DUMP START

                                .SBTTL  BASIC DEFINITIONS

                                ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
                        001100  STACK=  1100
                                .EQUIV  EMT,ERROR       ;;BASIC DEFINITION OF ERROR CALL
                                .EQUIV  IOT,SCOPE       ;;BASIC DEFINITION OF SCOPE CALL

                                ;*MISCELLANEOUS DEFINITIONS
                        000011  HT=     11              ;;CODE FOR HORIZONTAL TAB
                        000012  LF=     12              ;;CODE FOR LINE FEED
                        000015  CR=     15              ;;CODE FOR CARRIAGE RETURN
                        000200  CRLF=   200             ;;CODE FOR CARRIAGE RETURN-LINE FEED
                        177776  PS=     177776          ;;PROCESSOR STATUS WORD
                                .EQUIV  PS,PSW
                        177774  STKLMT=  177774         ;;STACK LIMIT REGISTER
                        177772  PIRQ=    177772         ;;PROGRAM INTERRUPT REQUEST REGISTER
                        177570  DSWR=    177570         ;;HARDWARE SWITCH REGISTER
                        177570  DDISP=   177570         ;;HARDWARE DISPLAY REGISTER

                                ;*GENERAL PURPOSE REGISTER DEFINITIONS
                        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
                        000006  SP=     %6              ;;STACK POINTER
                        000007  PC=     %7              ;;PROGRAM COUNTER
```

```
                                ;*PRIORITY LEVEL DEFINITIONS
000000                          PR0=    0               ;;PRIORITY LEVEL 0
000040                          PR1=    40              ;;PRIORITY LEVEL 1
000100                          PR2=    100             ;;PRIORITY LEVEL 2
000140                          PR3=    140             ;;PRIORITY LEVEL 3
000200                          PR4=    200             ;;PRIORITY LEVEL 4
000240                          PR5=    240             ;;PRIORITY LEVEL 5
000300                          PR6=    300             ;;PRIORITY LEVEL 6
000340                          PR7=    340             ;;PRIORITY LEVEL 7

                                ;*"SWITCH REGISTER" SWITCH DEFINITIONS
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

                                ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
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

                                        ;*BASIC "CPU" TRAP VECTOR ADDRESSES
                        000004          ERRVEC= 4               ;;TIME OUT AND OTHER ERRORS
                        000010          RESVEC= 10              ;;RESERVED AND ILLEGAL INSTRUCTIONS
                        000014          TBITVEC=14              ;;"T" BIT
                        000014          TRTVEC= 14              ;;TRACE TRAP
                        000014          BPTVEC= 14              ;;BREAKPOINT TRAP (BPT)
                        000020          IOTVEC= 20              ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
                        000024          PWRVEC= 24              ;;POWER FAIL
                        000030          EMTVEC= 30              ;;EMULATOR TRAP (EMT) **ERROR**
                        000034          TRAPVEC=34              ;;"TRAP" TRAP
                        000060          TKVEC=  60              ;;TTY KEYBOARD VECTOR
                        000064          TPVEC=  64              ;;TTY PRINTER VECTOR
                        000240          PIRQVEC=240             ;;PROGRAM INTERRUPT REQUEST VECTOR

                                        .SBTTL  ACT11 HOOKS

                                        ;;**************************************************************
                                        ;HOOKS REQUIRED BY ACT11
                        000224                  $SVPC=.                 ;SAVE PC
                        000046                  .=46
                000046  006626                  $ENDAD                  ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
                        000052                  .=52
                000052  000000                  .WORD   0               ;;2)SET LOC.52 TO ZERO
                        000224                  .=$SVPC                 ;; RESTORE PC
                        001000                  .=1000
                                        .SBTTL  APT PARAMETER BLOCK

                                        ;;**************************************************************
                                        ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
                                        ;;**************************************************************
                        001000                  .$X=.   ;;SAVE CURRENT LOCATION
                        000024                  .=24    ;;SET POWER FAIL TO POINT TO START OF PROGRAM
                000024  000200                  200     ;;FOR APT START UP
                        000044                  .=44    ;;POINT TO APT INDIRECT ADDRESS PNTR.
                000044  001000                  $APTHDR ;;POINT TO APT HEADER BLOCK
                        001000                  .=.$X   ;;RESET LOCATION COUNTER
                                        ;;**************************************************************
                                        ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
                                        ;INTERFACE SPEC.

                001000                  $APTHD:
```

```
  678  001000  000000                SHIBTS: .WORD   0          ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
  679  001002  001174                SMBADR: .WORD   $MAIL      ;;ADDRESS OF APT MAILBOX (BITS 0-15)
  680  001004  000055                SISTM:  .WORD   45.        ;;RUN TIM OF LONGEST TEST
  681  001006  000005                SPASTM: .WORD   5.         ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
  682  001010  000005                SUNITM: .WORD   5.         ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
  683  001012  000032                        .WORD   $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
```

```
                                .SBTTL  COMMON TAGS

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

                        001100          .=1100
001100                  SCMTAG:                 ;;START OF COMMON TAGS
001100  000000          STSTNM: .WORD   0       ;;CONTAINS THE TEST NUMBER
001102      000         SERFLG: .BYTE   0       ;;CONTAINS ERROR FLAG
001103      000         SICNT:  .BYTE   0       ;;CONTAINS SUBTEST ITERATION COUNT
001104  000000          SLPADR: .WORD   0       ;;CONTAINS SCOPE LOOP ADDRESS
001106  000000          SLPERR: .WORD   0       ;;CONTAINS SCOPE RETURN FOR ERRORS
001110  000000          SERTTL: .WORD   0       ;;CONTAINS TOTAL ERRORS DETECTED
001112      000         SITEMB: .BYTE   0       ;;CONTAINS ITEM CONTROL BYTE
001113      001         SERMAX: .BYTE   1       ;;CONTAINS MAX. ERRORS PER TEST
001114  000000          SERRPC: .WORD   0       ;;CONTAINS PC OF LAST ERROR INSTRUCTION
001116  000000          SGDADR: .WORD   0       ;;CONTAINS ADDRESS OF 'GOOD' DATA
001120  000000          SBDADR: .WORD   0       ;;CONTAINS ADDRESS OF 'BAD' DATA
001122  000000          SGDDAT: .WORD   0       ;;CONTAINS 'GOOD' DATA
001124  000000          SBDDAT: .WORD   0       ;;CONTAINS 'BAD' DATA
001126  000000                  .WORD   0       ;;RESERVED--NOT TO BE USED
001130  000000                  .WORD   0
001132      000         SAUTOB: .BYTE   0       ;;AUTOMATIC MODE INDICATOR
001133      000         SINTAG: .BYTE   0       ;;INTERRUPT MODE INDICATOR
001134  000000                  .WORD   0
001136  177570          SWR:    .WORD   DSWR    ;;ADDRESS OF SWITCH REGISTER
001140  177570          DISPLAY: .WORD  DDISP   ;;ADDRESS OF DISPLAY REGISTER
001142  177560          STKS:   177560          ;;TTY KBD STATUS
001144  177562          STKB:   177562          ;;TTY KBD BUFFER
001146  177564          STPS:   177564          ;;TTY PRINTER STATUS REG. ADDRESS
001150  177566          STPB:   177566          ;;TTY PRINTER BUFFER REG. ADDRESS
001152      000         SNULL:  .BYTE   0       ;;CONTAINS NULL CHARACTER FOR FILLS
001153      002         SFILLS: .BYTE   2       ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
001154      012         SFILLC: .BYTE   12      ;;INSERT FILL CHARS. AFTER A "LINE FEED"
001155      000         STPFLG: .BYTE   0       ;;"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
001156  000000          STIMES: 0               ;;MAX. NUMBER OF ITERATIONS
001160  000000          SESCAPE:0               ;;ESCAPE ON ERROR ADDRESS
001162  177607  000377  SBELL:  .ASCIZ  <207><377><377> ;;CODE FOR BELL
001166      077         SQUES:  .ASCII  /?/     ;;QUESTION MARK
001167      015         SCRLF:  .ASCII  <15>    ;;CARRIAGE RETURN
001170  000012          SLF:    .ASCIZ  <12>    ;;LINE FEED
                                ;;*****************************************************************
                                .SBTTL  APT MAILBOX-ETABLE

                                ;;*****************************************************************
                                .EVEN
001174                  SMAIL:                  ;;APT MAILBOX
001174  000000          SMSGTY: .WORD   AMSGTY  ;;MESSAGE TYPE CODE
001176  000000          SFATAL: .WORD   AFATAL  ;;FATAL ERROR NUMBER
001200  000000          STESTN: .WORD   ATESTN  ;;TEST NUMBER
001202  000000          SPASS:  .WORD   APASS   ;;PASS COUNT
001204  000000          SDEVCT: .WORD   ADEVCT  ;;DEVICE COUNT
```

```
   738  001206  000000                  $UNIT:  .WORD   AUNIT   ;;I/O UNIT NUMBER
   739  001210  000000                  $MSGAD: .WORD   AMSGAD  ;;MESSAGE ADDRESS
   740  001212  000000                  $MSGLG: .WORD   AMSGLG  ;;MESSAGE LENGTH
   741  001214                          SETABLE:                ;;APT ENVIRONMENT TABLE
   742  001214     000                  $ENV:   .BYTE   AENV    ;;ENVIRONMENT BYTE
   743  001215     000                  $ENVM:  .BYTE   AENVM   ;;ENVIRONMENT MODE BITS
   744  001216  000000                  $SWREG: .WORD   ASWREG  ;;APT SWITCH REGISTER
   745  001220  000000                  $USWR:  .WORD   AUSWR   ;;USER SWITCHES
   746  001222  000000                  $CPUOP: .WORD   ACPUOP  ;;CPU TYPE,OPTIONS
   747                                  ;*                      BITS 15-11=CPU TYPE
   748                                  ;*                           11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
   749                                  ;*                           11/70=06,PDQ=07,Q=10
   750                                  ;*                      BIT 10=REAL TIME CLOCK
   751                                  ;*                      BIT  9=FLOATING POINT PROCESSOR
   752                                  ;*                      BIT  8=MEMORY MANAGEMENT
   753  001224     000                  $MAMS1: .BYTE   AMAMS1  ;;HIGH ADDRESS,M.S. BYTE
   754  001225     000                  $MTYP1: .BYTE   AMTYP1  ;;MEM. TYPE,BLK#1
   755                                  ;*                      MEM.TYPE BYTE   --  (HIGH BYTE)
   756                                  ;*                           900 NSEC CORE=001
   757                                  ;*                           300 NSEC BIPOLAR=002
   758                                  ;*                           500 NSEC MOS=003
   759  001226  000000                  $MADR1: .WORD   AMADR1  ;;HIGH ADDRESS,BLK#1
   760                                  ;*                       MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
   761  001230     000                  $MAMS2: .BYTE   AMAMS2  ;;HIGH ADDRESS,M.S. BYTE
   762  001231     000                  $MTYP2: .BYTE   AMTYP2  ;;MEM.TYPE,BLK#2
   763  001232  000000                  $MADR2: .WORD   AMADR2  ;;MEM.LAST ADDRESS,BLK#2
   764  001234     000                  $MAMS3: .BYTE   AMAMS3  ;;HIGH ADDRESS,M.S.BYTE
   765  001235     000                  $MTYP3: .BYTE   AMTYP3  ;;MEM.TYPE,BLK#3
   766  001236  000000                  $MADR3: .WORD   AMADR3  ;;MEM.LAST ADDRESS,BLK#3
   767  001240     000                  $MAMS4: .BYTE   AMAMS4  ;;HIGH ADDRESS,M.S.BYTE
   768  001241     000                  $MTYP4: .BYTE   AMTYP4  ;;MEM.TYPE,BLK#4
   769  001242  000000                  $MADR4: .WORD   AMADR4  ;;MEM.LAST ADDRESS,BLK#4
   770  001244  000000                  $VECT1: .WORD   AVECT1  ;;INTERRUPT VECTOR#1,BUS PRIORITY#1
   771  001246  000000                  $VECT2: .WORD   AVECT2  ;;INTERRUPT VECTOR#2BUS PRIORITY#2
   772  001250  020000                  $BASE:  .WORD   ABASE   ;;BASE ADDRESS OF EQUIPMENT UNDER TEST
   773  001252  000000                  $DEVM:  .WORD   ADEVM   ;;DEVICE MAP
   774  001254  140000                  $CDW1:  .WORD   ACDW1   ;;CONTROLLER DESCRIPTION WORD#1
   775  001256  000004                  $CDW2:  .WORD   ACDW2   ;;CONTROLLER DESCRIPTION WORD#2
   776  001260                          SETEND:
   777                                  .MEXIT
```

```
                                        .SBTTL  ERROR POINTER TABLE

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

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


001260                                  $ERRTB:
                                        ;ITEM1
001260  013436  013464  013516                  EM1,DH1,DT1,DF          ;READ-WRITE ERROR
001266  015110
                                        ;ITEM2
001270  013532  013555  013610                  EM2,DH2,DT2,DF          ;BASIC ADDRESS TEST.
001276  015110
                                        ;ITEM3
001300  013624  013651  013716                  EM3,DH3,DT3,DF          ;FAST READ TEST.
001306  015110
                                        ;ITEM4
001310  013734  014002  014050                  EM4,DH4,DT4,DF          ;SLIDE TEST. ADDR-2 ALTERED.
001316  015110
                                        ;ITEM5
001320  014070  014136  014204                  EM5,DH5,DT5,DF          ;SLIDE TEST. ADDR+2 ALTERED.
001326  015110
                                        ;ITEM6
001330  014224  014265  014320                  EM6,DH6,DT6,DF          ;ALTERNATE BIT ERROR.
001336  015110
                                        ;ITEM7
001340  014334  014361  014420                  EM7,DH7,DT7,DF          ;RAM BYTE TEST ERROR
001346  015110
                                        ;ITEM10
001350  014436  014457  014512                  EM10,DH10,DT10,DF       ;PROM SPECIAL DATA TEST READ ERROR
001356  015110
                                        ;ITEM11
001360  014526  014553  014624                  EM11,DH11,DT11,DF       ;PROM TO RAM COPY ERROR.
001366  015110
                                        ;ITEM12
001370  014642  014671  014730                  EM12,DH12,DT12,DF       ;PROM WRITE TRAP FAILURE.
001376  015110
                                        ;ITEM13
001400  014744  014777  015024                  EM13,DH13,DT13,DF       ;BUS TIME-OUT (TRAP) ERROR.
001406  015110
                                        ;ITEM14
001410  015034  000000  000000                  EM14,0,0,0              ;MEM MAP ERROR.
001416  000000
                                                                        ;NON-STANDARD MAP SIGNATURE.
                                                                        ;DH, DT, AND DF = 0
```

```
                                                .SBTTL  INITIAL START-UP.
001420  000000                                  RAM:    0                       ;RAM START LOC.
001422  000000                                  RAMEND: 0                       ;RAM LAST LOC
001424  000000                                  PRM:    0                       ;PROM START LOC
001426  000000                                  PRMEND: 0                       ;PROM LAST LOC
001430  000000                                  PRMSZ:  0                       ;PROM SIZE (IN K)
001432  000000                                  PRMX:   0                       ;0 NORM, -1 IF SPECIAL 4K SET.
001434  000000                                  MAINT:  0                       ;0 NORM, -1 IF ANY DUMP START.

001436                                          START:
                                                .SBTTL  INITIALIZE THE COMMON TAGS
                                                ;;CLEAR THE COMMON TAGS (SCMTAG) AREA
001436  012706  001100                                  MOV     #SCMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
001442  005026                                        1$: CLR     (R6)+           ;;CLEAR MEMORY LOCATION
001444  022706  001140                                  CMP     #SWR,R6 ;;DONE?
001450  001374                                          BNE     1$              ;;LOOP BACK IF NO
001452  012706  001100                                  MOV     #STACK,SP       ;;SETUP THE STACK POINTER
                                                ;;INITIALIZE A FEW VECTORS
001456  012737  007654  000020                        MOV     #$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
001464  012737  000340  000022                        MOV     #340,@#IOTVEC+2 ;;LEVEL 7
001472  012737  010214  000030                        MOV     #ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
001500  012737  000340  000032                        MOV     #340,@#EMTVEC+2 ;;LEVEL 7
001506  012737  013120  000034                        MOV     #TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
001514  012737  000340  000036                        MOV     #340,@#TRAPVEC+2;LEVEL 7
001522  012737  012474  000024                        MOV     #PWRON,@#PWRVEC ;;POWER FAILURE VECTOR
001530  012737  000340  000026                        MOV     #340,@#PWRVEC+2 ;;LEVEL 7
001536  013737  006574  006566                        MOV     @#ENDCT,@#EOPCT ;;SETUP END-OF-PROGRAM COUNTER
001544  005037  001160                                CLR     $TIMES          ;;INITIALIZE NUMBER OF ITERATIONS
001550  005037  001162                                CLR     $ESCAPE         ;;CLEAR THE ESCAPE ON ERROR ADDRESS
001554  112737  000001  001115                        MOVB    #1,$ERRMAX      ;;ALLOW ONE ERROR PER TEST
001562  012737  001562  001106                        MOV     #.,$LPADR       ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
001570  012737  001570  001110                        MOV     #.,$LPERR       ;;SETUP THE ERROR LOOP ADDRESS
                                                ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
                                                ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
001576  013746  000004                                  MOV     @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
001602  012737  001636  000004                        MOV     #64$,@#ERRVEC   ;;SET UP ERROR VECTOR
001610  012737  177570  001140                        MOV     #DSWR,SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
001616  012737  177570  001142                        MOV     #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
001624  022777  177777  177306                        CMP     #-1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
001632  001012                                        BNE     66$             ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
                                                                                ;;AND  THE HARDWARE SWR IS NOT = -1
001634  000403                                        BR      65$             ;;BRANCH IF NO TIMEOUT
001636  012716  001644                          64$:  MOV     #65$,(SP)       ;;SET UP FOR TRAP RETURN
001642  000002                                        RTI
001644  012737  000176  001140                  65$:  MOV     #SWREG,SWR      ;;POINT TO SOFTWARE SWR
001652  012737  000174  001142                        MOV     #DISPREG,DISPLAY
001660  012637  000004                          66$:  MOV     (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR

001664  005037  001202                                CLR     $PASS           ;;CLEAR PASS COUNT
001670  132737  000200  001215                        BITB    #APTSIZE,$ENVM  ;;TEST USER SIZE UNDER APT
001676  001403                                        BEQ     67$             ;;YES,USE NON-APT SWITCH
001700  012737  001216  001140                        MOV     #$SWREG,SWR     ;;NO,USE APT SWITCH REGISTER
001706                                          67$:
```

```
001706  005737  001434                  TST     MAINT           ;MAINT FLAG SET ??
001712  001401                          BEQ     .+4             ;NO, PROCEED WITH NORMAL START.
001714  000207                          RTS     PC              ;YES, RETURN TO MAINT ROUTINE.

                                .SBTTL  TYPE PROGRAM NAME
                                ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
001716  005227  177777                  INC     #-1             ;;FIRST TIME?
001722  001054                          BNE     68$             ;;BRANCH IF NO
001724  022737  006626  000042          CMP     #$ENDAD,@#42    ;;ACT-11?
001732  001450                          BEQ     68$             ;;BRANCH IF YES
001734  104401  002002                  TYPE    ,63$            ;;TYPE ASCIZ STRING
                                .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
001740  005737  000042                  TST     @#42            ;;ARE WE RUNNING UNDER XXDP/ACT?
001744  001012                          BNE     70$             ;;BRANCH IF YES
001746  123727  001214  000001          CMPB    $ENV,#1         ;;ARE WE RUNNING UNDER APT?
001754  001406                          BEQ     70$             ;;BRANCH IF YES
001756  023727  001140  000176          CMP     SWR,#SWREG      ;;SOFTWARE SWITCH REG SELECTED?
001764  001005                          BNE     71$             ;;BRANCH IF NO
001766  104407                          GTSWR                   ;;GET SOFT-SWR SETTINGS
001770  000403                          BR      71$
001772  112737  000001  001134  70$:    MOVB    #1,$AUTOB       ;;SET AUTO-MODE INDICATOR
002000                          71$:
002000  000425                          BR      68$             ;;GET OVER THE ASCIZ
                                ;;63$:  .ASCIZ  <CRLF>/MD-11-DVMRA-A.  LSI-11 UVPROM-RAM TEST./<CRLF>
002054                          68$:
                                .SBTTL  INITIALIZE RAM AND PROM ADDRESSES.
                                ;IF APT, ACT, OR XXDP, SKIP DIALOGUE AND USE DEFAULTS.
                                ;RAM = 20000, PROM = 140000, SIZE = 4K.
                                ;
002054  005737  000042                  TST     @#42            ;ACT OR XXDP ??
002060  001004                          BNE     1$              ;YES, USE DEFAULTS
002062  123727  001214  000001          CMPB    $ENV,#1         ;NO, HOW 'BOUT APT ??
002070  001044                          BNE     INIT1           ;NO, INITIALIZE VIA OPERATOR

002072  013737  001250  001420  1$:     MOV     $BASE,RAM       ;ACT, XXDP, OR APT, USE DEFAULTS
002100  013737  001250  001422          MOV     $BASE,RAMEND    ;...1ST RAM LOC = 20000
002106  062737  000776  001422          ADD     #776,RAMEND     ;...LAST RAM LOC = 020776

002114  005037  001432                  CLR     PRMX            ;CLEAR SPECIAL PROM FLAG
002120  013737  001254  001424          MOV     $CSW1,PRM       ;...1ST PROM LOC = 140000
002126  013737  001256  001430          MOV     $CSW2,PRMSZ     ;...SIZE = 4K
002134  013701  001256                  MOV     $CSW2,R1        ;...CALCULATE LAST LOC.
002140  012737  003776  001426          MOV     #3776,PRMEND    ;...IN PRMEND
002146  005301                  2$:     DEC     R1
002150  001404                          BEQ     3$
002152  062737  004000  001426          ADD     #4000,PRMEND
002160  000772                          BR      2$
002162  063737  001424  001426  3$:     ADD     PRM,PRMEND      ;LAST PROM LOC = 1ST + K.-2
002170  012737  000377  007022          MOV     #377,MAPSW      ;INH TEST 1 MAP OUTPUT.
002176  000137  002620                  JMP     RSTRT           ;GO START 'EM UP.
```

```
                                        ;INITIALIZE RAM ADDRESSES.  IF 0, USE DEFAULT ADDRESS.
                                        ;
002202                                  INIT1:
002202  104401  002210                          TYPE    ,65$            ;;TYPE ASCIZ STRING
002206  000410                                  BR      64$             ;;GET OVER THE ASCIZ
                                        ;;65$:  .ASCIZ  <CRLF>/RAM ADDRESS = /
002230                                  64$:
002230  104413                                  RDOCT
002232  012637  001420                          MOV     (SP)+,RAM       ;;POP STACK INTO RAM
002236  001003                                  BNE     1$              ;IS IT 0 ??
002240  013737  001250  001420                  MOV     $BASE,RAM       ;YES, USE DEFAULT RAM (20000)
002246  033727  001420  000777          1$:     BIT     RAM,#777        ;256. WORD (1000 BYTE) BOUNDARY ??
002254  001013                                  BNE     2$              ;NO, COMPLAIN
002256  013701  001420                          MOV     RAM,R1          ;R1=START LOC
002262  012702  000776                          MOV     #776,R2
002266  060102                                  ADD     R1,R2           ;R2=LAST LOC
002270  010237  001422                          MOV     R2,RAMEND       ;SAVE IT
002274  004737  006662                          JSR     PC,NXM          ;TEST BOTH FOR NXM
002300  000401                                  BR      2$              ;ONE OR T'OTHER TRAPPED !!!!!
002302  000403                                  BR      INIT2           ;OK, GO INIT PROM SIZE.
                                        ;
002304  104401  013204                  2$:     TYPE    ,M1             ;SAY "INVALID ADDRESS"...
002310  000734                                  BR      INIT1           ;...AND TRY AGAIN.
                                        ;
                                        ;NOW GET THE PROM SIZE.  IF 0, DEFAULT TO 4K.
                                        ;IF X, SPECIAL 4K SET INSTALLED.  OTHERWISE ACCEPT 1-4 ONLY.
                                        ;ALSO ACCEPT 1024, 2048, 3072, OR 4096, ALTHO ACTUALLY USE
                                        ;JUST THE FIRST DIGIT.
                                        ;
002312                                  INIT2:
002312  104401  002320                          TYPE    ,65$            ;;TYPE ASCIZ STRING
002316  000407                                  BR      64$             ;;GET OVER THE ASCIZ
                                        ;;65$:  .ASCIZ  <CRLF>/PROM SIZE = /
002336                                  64$:
002336  104412                                  RDLIN
002340  012600                                  MOV     (SP)+,R0        ;POP ADDR OF 1ST CHAR.
002342  111037  001430                          MOVB    (R0),PRMSZ      ;GET THAT CHARACTER.
002346  105037  001431                          CLRB    PRMSZ+1         ;CLEAR HI BYTE.
002352  005037  001432                          CLR     PRMX            ;CLEAR SPECIAL PROM FLAG
002356  005737  001430                          TST     PRMSZ           ;SIZE = 0 (CR ONLY) ??
002362  001004                                  BNE     1$              ;NO IT'S NOT
002364  013737  001256  001430                  MOV     $CDW2,PRMSZ     ;YES, DEFAULT IS 4K.
002372  000432                                  BR      INIT3           ;GO GET START ADDRESS.
                                        ;
002374  022737  000130  001430          1$:     CMP     #'X,PRMSZ       ;SIZE = "X" ??
002402  001007                                  BNE     2$              ;NO.
002404  012737  177777  001432                  MOV     #-1,PRMX        ;YES, SET SPECIAL FLAG...
002412  012737  000004  001430                  MOV     #4,PRMSZ        ;...AND SET SIZE = 4K.
002420  000417                                  BR      INIT3           ;GO GET ADDRESS.
                                        ;
002422  162737  000060  001430          2$:     SUB     #60,PRMSZ       ;STRIP ASCII BITS
002430  100410                                  BMI     3$              ;SIZE < 0, INVALID.
002432  162737  000005  001430                  SUB     #5,PRMSZ
002440  100004                                  BPL     3$              ;SIZE > 4, INVALID.
```

```
                                        ADD     #5,PRMSZ        ;SIZE = 1, 2, 3, OR 4.
                                        BR      INIT3           ;NOW GET PROM ADDRESS

002452  104401  001170        3$:     TYPE    ,$QUES          ;TYPE '?'...
002456  000715                        BR      INIT2           ;...AND TRY AGAIN.

                                ;
                                ;NOW INITIALIZE THE PROM ADDRESSES.  IF 0, USE DEFAULT.
                                ;
002460                          INIT3:
002460  104401  002466                TYPE    ,65$            ;;TYPE ASCIZ STRING
002464  000411                        BR      64$             ;;GET OVER THE ASCIZ
                                ;;65$:  .ASCIZ  <CRLF>/PROM ADDRESS = /
002510                          64$:
002510  104413                        RDOCT
002512  012637  001424                MOV     (SP)+,PRM       ;;POP STACK INTO PRM
002516  001003                        BNE     1$              ;IS IT 0 ??
002520  013737  001254  001424        MOV     $CDW1,PRM       ;YES, USE DEFAULT PROM (140000).
002526  032737  003777  001424  1$:   BIT     #3777,PRM       ;ON 1K WORD (4000 BYTE) BOUNDARY ??
002534  001025                        BNE     4$              ;NO, COMPLAIN.

002536  013701  001424                MOV     PRM,R1          ;YES, SET UP FOR NXM CHECK.
002542  012702  003776                MOV     #3776,R2        ;DETERMINE LAST ADDRESS...
002546  013703  001430                MOV     PRMSZ,R3        ;...BASED ON PROM SIZE.
002552  005303                  2$:     DEC     R3
002554  001403                        BEQ     3$              ;R2 = SIZE-2
002556  062702  004000                ADD     #4000,R2
002562  000773                        BR      2$

002564  060102                  3$:     ADD     R1,R2           ;R2 = LAST PROM ADDRESS.
002566  010237  001426                MOV     R2,PRMEND       ;SAVE IT.
002572  004737  006662                JSR     PC,NXM          ;CHECK BOTH ADDRESSES.
002576  000404                        BR      4$              ;ONE OF THEM TRAPPED !!!!
002600  005037  007022                CLR     MAPSW           ;ENABLE TEST 1 MAP OUTPUT.
002604  000137  002620                JMP     RSTRT           ;OK, GO START TESTING.

002610  104401  013301        4$:     TYPE    ,M2             ;SAY "INVALID PROM RANGE ETC"
002614  000137  002312                JMP     INIT2           ;...AND TRY AGAIN.
```

```
1025  002620  005037  001202          RSTRT:  CLR     $PASS           ;CLEAR PASS COUNTER
1026  002624  012706  001100          AGAIN:  MOV     #STACK,SP       ;INIT THE SP. (LOOP HERE ON EOP)
1027                                  ;;*****************************************************************
1028                                  ;*TEST 1         BUILD AND TEST A 32K MEMORY MAP.
1029                                  ;;*****************************************************************
1030  002630  000240                  TST1:   NOP
1031  002632  012737  000100  001160          MOV     #100,$TIMES     ;;DO 100 ITERATIONS
1032  002640  112737  000001  001102          MOVB    #1,$TSTNM       ;INIT TEST NUMBER
1033  002646  012737  000001  001200          MOV     #1,$TESTN       ;APT TOO
1034                                          .ENABL  LSB
1035  002654  012737  003146  001106          MOV     #7$,$LPADR      ;SET LOOP ADDRESS
1036  002662  013737  001106  001110          MOV     $LPADR,$LPERR   ;AND ERROR LOOPER.
1037                                  ;
1038                                  ;ENTER AT TST1A FROM "MDUMP" START. DUMP THE MAP AND RETURN.
1039                                  ;
1040  002670  005037  007020          TST1A:  CLR     LOC             ;START AT LOCATION 0 (0-4K).
1041  002674  012737  000020  007016          MOV     #16.,MAPK       ;INIT SEGMENT COUNT
1042  002702  012737  007024  007012          MOV     #MEMMAP,RMAP    ;...R MAP POINTER...
1043  002710  012737  007026  007014          MOV     #MEMMAP+2,WMAP  ;...AND W MAP POINTER.
1044
1045  002716  105737  007022                  TSTB    MAPSW           ;LO BYTE INHIBITS OUTPUT IF AUTO
1046  002722  001004                          BNE     1$
1047  002724  104401  001171                  TYPE    ,$CRLF          ;DOUBLE CRLF.
1048  002730  104401  001171                  TYPE    ,$CRLF
1049  002734  017700  004060          1$:     MOV     @LOC,R0         ;READ..., AND IF NO-TRAP...
1050  002740  010077  004054          RD1:    MOV     R0,@LOC         ;...WRITE, AND IF NO-TRAP...
1051  002744  000261                  WRT1:   SEC                     ;...YOU'RE HERE. "LOC" IS R/W.
1052  002746  006177  004040                  ROL     @RMAP           ;SET BIT IN READ MAP...
1053  002752  000261                          SEC
1054  002754  006177  004034                  ROL     @WMAP           ;...AND IN WRITE MAP.
1055  002760  105737  007022                  TSTB    MAPSW
1056  002764  001032                          BNE     2$
1057  002766  104401  015120                  TYPE    ,W              ;W = READ/WRITE SEGMENT.
1058  002772  000427                          BR      2$
1059
1060  002774  000241          RTN1:   CLC                     ;YOU'RE HERE IF "LOC" WAS NEXM.
1061  002776  006177  004010                  ROL     @RMAP           ;CLEAR BIT IN READ MAP...
1062  003002  000241                          CLC
1063  003004  006177  004004                  ROL     @WMAP           ;...AND WRITE MAP.
1064  003010  105737  007022                  TSTB    MAPSW
1065  003014  001016                          BNE     2$
1066  003016  104401  015114                  TYPE    ,DASH           ;DASH = NEXM SEGMENT.
1067  003022  000413                          BR      2$
1068
1069  003024  000261          RTN2:   SEC                     ;YOU'RE HERE IF "LOC" WAS READ ONLY.
1070  003026  006177  003760                  ROL     @RMAP           ;SET BIT IN READ MAP...
1071  003032  000241                          CLC
1072  003034  006177  003754                  ROL     @WMAP           ;CLEAR IT IN WRITE MAP.
1073  003040  105737  007022                  TSTB    MAPSW
1074  003044  001002                          BNE     2$
1075  003046  104401  015116                  TYPE    ,R              ;R = READ ONLY SEGMENT.
```

```
1076  003052  005337  007016          2$:     DEC     MAPK            ;BUMP SEGMENT COUNT
1077  003056  001016                          BNE     3$              ;BR IF 4K (16BITS) NOT DONE YET.
1078  003060  012737  000020  007016          MOV     #16.,MAPK       ;INIT COUNT FOR NEXT BANK
1079  003066  062737  000004  007012          ADD     #4,RMAP         ;BUMP MAP POINTERS
1080  003074  062737  000004  007014          ADD     #4,WMAP
1081
1082  003102  105737  007022                  TSTB    MAPSW
1083  003106  001002                          BNE     3$
1084  003110  104401  001171                  TYPE    ,$CRLF          ;CRLF IF NOT INHIBITED
1085  003114  062737  001000  007020  3$:     ADD     #1000,LOC       ;INIT POINTER FOR NEXT SEGMENT
1086  003122  001304                          BNE     1$              ;CONTINUE...
1087                                                                  ;..."LOC" WILL BE 0 AT 32K.
1088  003124  105737  007022                  TSTB    MAPSW
1089  003130  001006                          BNE     7$
1090  003132  104401  001171                  TYPE    ,$CRLF          ;FINAL CRLF IF NOT INHIBITED.
1091                                  ;
1092                                  ;NOW, ON INITIAL PASS, COPY MEMMAP TO SAVMAP.  ON SUBSEQUENT
1093                                  ;PASSES, COMPARE REFERENCE (SAVMAP) TO CURRENT (MEMMAP)
1094                                  ;AND REPORT ANY DISCREPANCIES.
1095                                  ;IF ENTRY WAS FROM "MDUMP" START, RETURN THERE WITHOUT
1096                                  ;ANY FURTHER ACTION.
1097                                  ;
1098  003136  005737  001434                  TST     MAINT           ;MAINT FLAG SET ??
1099  003142  001401                          BEQ     .+4             ;NO
1100  003144  000207                          RTS     PC              ;YES, RETURN TO "MDUMP".
1101
1102  003146  005000                  7$:     CLR     R0              ;INIT MAP INDEX
1103  003150  105737  007023                  TSTB    MAPSW+1         ;HI BYTE = 0, ON INIT PASS.
1104  003154  001013                          BNE     5$              ;NO, COMPARE THIS MAP WITH REF.
1105  003156  016060  007024  007064  4$:     MOV     MEMMAP(R0),SAVMAP(R0) ;YES, SAVE THE REF MAP.
1106  003164  005720                          TST     (R0)+
1107  003166  022700  000034                  CMP     #28.,R0         ;DONE ??
1108  003172  001371                          BNE     4$              ;NO
1109  003174  012737  177777  007022          MOV     #-1,MAPSW       ;YES, SET SWITCH TO INH OUTPUT...
1110  003202  000632                          BR      TST1A           ;...AND COMPARE FROM NOW ON.
1111
1112  003204  026060  007024  007064  5$:     CMP     MEMMAP(R0),SAVMAP(R0) ;COMPARE MAPS.
1113  003212  001005                          BNE     10$             ;ERROR
1114  003214  005720                  6$:     TST     (R0)+           ;CONTINUE AFTER ERROR
1115  003216  022700  000034                  CMP     #28.,R0         ;DONE ??
1116  003222  001370                          BNE     5$              ;NOT YET
1117  003224  000456                          BR      TST2            ;;DONE, GO NEXT TEST.
```

```
1118                                    ;;*****************************************************************
1119                                    ;MAP ERROR.  RE-CONSTRUCT AND OUTPUT A MAP FOR THE FAILING
1120                                    ;BANK.  CONTINUE AFTER OUTPUT.
1121                                    ;
1122  003226  010001                   10$:    MOV     R0,R1           ;INDEX/2 + 1 = BANK NUMBER
1123  003230  006201                           ASR     R1
1124  003232  006201                           ASR     R1
1125  003234  062701  000061                   ADD     #61,R1          ;R1 = ASCII BANK #.
1126  003240  110137  015064                   MOVB    R1,EM14A        ;SET # IN ERROR MSG STRING.
1127
1128  003244  032700  000002                   BIT     #BIT1,R0        ;ADJUST INDEX TO PICK 1ST WORD OF PAIR.
1129  003250  001401                           BEQ     .+4             ;NO ADJ REQ'D.
1130  003252  005740                           TST     -(R0)           ;DEC INDEX
1131  003254  016037  007024  001122           MOV     MEMMAP(R0),SBOADR ;FAILING R WORD
1132  003262  005720                           TST     (R0)+           ;BUMP INDEX
1133  003264  016037  007024  001126           MOV     MEMMAP(R0),SBODAT ;FAILING W WORD
1134
1135  003272  005001                           CLR     R1              ;INIT BIT COUNTER
1136  003274  005737  001122           11$:    TST     $BOADR
1137  003300  100007                           BPL     12$             ;IF R BIT = 0, NEXM.
1138  003302  005737  001126                   TST     $BODAT          ;IF R BIT = 1...
1139  003306  100010                           BPL     13$             ;...AND W = 0, READ ONLY.
1140                                                                   ;IF BOTH = 1, READ/WRITE.
1141  003310  112761  000127  015067           MOVB    #'W,EM14B(R1)   ;SET APPROPRIATE CHARACTER...
1142  003316  000407                           BR      14$
1143  003320  112761  000055  015067  12$:     MOVB    #'-,EM14B(R1)   ;. .IN ERROR MAP...
1144  003326  000403                           BR      14$
1145  003330  112761  000122  015067  13$:     MOVB    #'R,EM14B(R1)   ;...CHARACTER STRING.
1146
1147  003336  006137  001122           14$:    ROL     $BOADR          ;NOW ROTATE TO NEXT BIT
1148  003342  006137  001126                   ROL     $BODAT
1149  003346  005201                           INC     R1              ;BUMP BIT COUNT (INDEX)
1150  003350  022701  000020                   CMP     #16.,R1
1151  003354  001347                           BNE     11$             ;CONTINUE FOR 16 BITS.
1152  003356  104014                           ERROR   14              ;PRINT ERROR "ITEM 14"...
1153  003360  000715                           BR      6$              ;...AND CONTINUE TEST.
1154                                    ;;*****************************************************************
1155                                            .DSABL  LSB
```

```
                                ;;*****************************************************************
                                ;TEST 2         RAM -- BASIC 1'S AND 0'S TEST.
                                ;;*****************************************************************
003362  000004                  TST2:   SCOPE
003364  012737  000100  001160          MOV     #100,STIMES     ;;DO 100 ITERATIONS
003372  005037  003416                  CLR     PATRN           ;FIRST PATTERN IS ZEROS
003376  004737  003420                  JSR     PC,DATTST       ;RUN IT...
003402  012737  177777  003416          MOV     #-1,PATRN       ;NEXT PATTERN IS ONES.
003410  004737  003420                  JSR     PC,DATTST       ;DO IT...
003414  000453                          BR      TST3            ;;SKIP OVER SUBRTN TO NEXT TEST.
                                ;
                                ;SUBROUTINE FOR 1'S AND 0'S TEST.
                                ;
003416  000000                  PATRN:  0
003420  012701  000400          DATTST: MOV     #256.,R1        ;WORD COUNT
003424  013702  001420                  MOV     RAM,R2          ;FIRST ADDRESS
003430  013722  003416          1$:     MOV     PATRN,(R2)+     ;FILL THE RAM.
003434  005301                          DEC     R1
003436  001374                          BNE     1$              ;CONTINUE TIL DONE.
003440  012701  000400                  MOV     #256.,R1        ;OK, RESET COUNT.
003444  012737  003532  001110          MOV     #44$,$LPERR     ;SET FOR SWR9 OPTION IN 'ERROR'
003452  013702  001420                  MOV     RAM,R2          ;SET FIRST ADDRESS.
003456  022722  003416          2$:     CMP     PATRN,(R2)+     ;DATA OK ??
003462  001006                          BNE     4$              ;NO, DATA ERROR.
003464  005301                  3$:     DEC     R1              ;YES.
003466  001373                          BNE     2$              ;CONTINUE TIL DONE
003470  013737  001106  001110          MOV     $LPADR,$LPERR   ;SET FOR SWR9 OPTION IN 'SCOPE'
003476  000207                          RTS     PC              ;RETURN TO CALLER
                                ;;*****************************************************************
003500  013737  003416  001124  4$:     MOV     PATRN,$GOODT    ;GOOD DATA.
003506  016237  177776  001126          MOV     -2(R2),$BDDAT   ;BAD DATA
003514  010237  001122                  MOV     R2,$BDADR
003520  062737  177776  001122          ADD     #-2,$BDADR      ;BAD ADDRESS
003526  104001                          ERROR   1               ;PRINT ERROR "ITEM 1".
003530  000755                          BR      3$              ;CONTINUE AFTER ERROR. (SWR9=0)
003532  062702  177776          44$:    ADD     #-2,R2          ;LOOP ON THIS ERROR (SWR9=1)
003536  013712  003416                  MOV     PATRN,(R2)      ;REFRESH FAILED LOCATION...
003542  000745                          BR      2$              ;...AND TRY AGAIN.
                                ;;*****************************************************************
```

```
1200                                    ;;*****************************************************************
1201                                    ;*TEST 3        RAM -- BASIC ADDRESS TEST.
1202                                    ;;*****************************************************************
1203  003544  000004                    TST3:   SCOPE
1204  003546  012737  000100  001160            MOV     #100,$TIMES     ;;DO 100 ITERATIONS
1205  003554  012701  000400                    MOV     #256.,R1        ;WORD COUNT
1206  003560  013702  001420                    MOV     RAM,R2          ;FIRST ADDRESS
1207  003564  010212                    1$:     MOV     R2,(R2)         ;FILL WITH OWN ADDRESS.
1208  003566  005722                            TST     (R2)+           ;BUMP ADDRESS POINTER
1209  003570  005301                            DEC     R1              ;DONE YET ??
1210  003572  001374                            BNE     1$              ;NO, CONTINUE FILLING.
1211
1212  003574  012737  003654  001110            MOV     #44$,$LPERR     ;SET FOR SWR9 OPTION IN 'ERROR'
1213  003602  012701  000400                    MOV     #256.,R1        ;RESET COUNT
1214  003606  013702  001420                    MOV     RAM,R2          ;...AND ADDRESS.
1215  003612  020212                    2$:     CMP     R2,(R2)         ;CO?ARE EACH ADDRESS.
1216  003614  001007                            BNE     4$              ;ADDRESS ERROR
1217  003616  005722                    3$:     TST     (R2)+           ;BUMP ADDRESS
1218  003620  005301                            DEC     R1
1219  003622  001373                            BNE     2$              ;AND CONTINUE TIL DONE
1220
1221  003624  013737  001106  001110            MOV     $LPADR,$LPERR   ;SET FOR SWR9 OPTION IN 'SCOPE'
1222  003632  000412                            BR      TST4            ;;DONE, GO TO NEXT TEST.
1223
1224                                    ;;*****************************************************************
1225  003634  010237  001122            4$:     MOV     R2,$BDADR       ;BAD ADDRESS
1226  003640  010237  001124                    MOV     R2,$GDDAT       ;GOOD DATA
1227  003644  011237  001126                    MOV     (R2),$BDDAT     ;BAD DATA
1228  003650  104002                            ERROR   2               ;PRINT ERROR "ITEM 2".
1229  003652  000761                            BR      3$              ;CONTINUE AFTER ERROR. (SWR9=0)
1230                                                                    ;... OR ...
1231  003654  010212                    44$:    MOV     R2,(R2)         ;RETRY ON THIS ERROR (SWR9=1)
1232  003656  000755                            BR      2$
1233                                    ;;*****************************************************************
```

```
1234                                    ;;*****************************************************************
1235                                    ;*TEST 4        RAM -- FAST READ TEST.
1236                                    ;;*****************************************************************
1237  003660  000004                    TST4:   SCOPE
1238  003662  012737  000100  001160            MOV     #100,STIMES     ;;DO 100 ITERATIONS
1239  003670  013701  001420            .$:     MOV     RAM,R1          ;RAM START LOC.
1240  003674  012702  015122                    MOV     #IBUF,R2        ;INTERNAL BUFFER ADDRESS
1241  003700  013703  001422                    MOV     RAMEND,R3
1242  003704  062703  000002                    ADD     #2,R3           ;END OF RAM +2.
1243
1244  003710  012122                    1$:     MOV     (R1)+,(R2)+     ;FILL THE BUFFER WITH 256
1245                                                                    ;CONSECUTIVE READS.
1246                                    ;
1247                                    ;THE MOV INSTR ABOVE IS REPEATED 255 TIMES.
1248                                    ;LISTING TURNED OFF FOR ALL THAT.
1249                                    ;
1250                                    ;
1251                                    ;NOW THE INTERNAL BUFFER IS A RAM IMAGE.
1252                                    ;
1253  004710  013701  001420                    MOV     RAM,R1          ;RESET POINTERS...
1254  004714  012702  015122                    MOV     #IBUF,R2
1255  004720  022122                    .$:     CMP     (R1)+,(R2)+     ;...AND COMPARE EACH ADDRESS
1256  004722  001003                            BNE     2$              ;ADDRESS ERRROR.
1257
1258  004724  020103                    3$:     CMP     R1,R3           ;DONE YET ??
1259  004726  001374                            BNE     1$              ;NOPE, CONTINUE CHECKING.
1260  004730  000415                            BR      TST5            ;;YES, GO TO NEXT TEST.
1261
1262                                    ;;*****************************************************************
1263  004732  016137  177776  001122    2$:     MOV     -2(R1),$BOARDR  ;RAM WORD
1264  004740  016237  177776  001126            MOV     -2(R2),$BODAT   ;BUFFER WORD
1265  004746  010137  001120                    MOV     R1,$GOARDR
1266  004752  062737  177776  001120            ADD     #-2,$GOARDR     ;RAM ADDRESS (=GOOD DATA).
1267  004760  104003                            ERROR   3               ;PRINT ERROR "ITEM 3".
1268  004762  000760                            BR      3$              ;CONTINUE AFTER ERROR (SWR9=0)
1269                                                                    ;...OR RESTART AT 'TST3+2'  (SWR9=1)
1270                                    ;;*****************************************************************
```

```
1271                                    ;;*****************************************************************
1272                                    ;*TEST 5        RAM -- SLIDING 0 BIT TEST.
1273                                    ;;*****************************************************************
1274  004764  000004                    TST5:   SCOPE
1275  004766  012737  000100  001160            MOV     #100,$TIMES     ;;DO 100 ITERATIONS
1276  004774  012737  177777  005060            MOV     #-1,FILLR       ;BACKGROUND = 1'S
1277  005002  012737  177776  005062            MOV     #-2,SLIDR       ;SLIDE A 0 BIT.
1278  005010  004737  005064                    JSR     PC,SLIDE        ;GO DO IT.
1279  005014  013737  001106  001110            MOV     $LPADR,$LPERR   ;SET FOR SWR9 OPTION IN 'SCOPE'
1280
1281                                    ;;*****************************************************************
1282                                    ;*TEST 6        RAM -- SLIDING 1 BIT TEST.
1283                                    ;;*****************************************************************
1284  005022  000004                    TST6:   SCOPE
1285  005024  012737  000100  001160            MOV     #100,$TIMES     ;;DO 100 ITERATIONS
1286  005032  005037  005060                    CLR     FILLR           ;BACKGROUND = 0'S
1287  005036  012737  000001  005062            MOV     #1,SLIDR        ;SLIDE A 1 BIT.
1288  005044  004737  005064                    JSR     PC,SLIDE        ;GO DO IT.
1289  005050  013737  001106  001110            MOV     $LPADR,$LPERR   ;SET FOR SWR9 OPTION IN 'SCOPE'
1290  005056  000544                    BR      TST7            ;;DONE, SKIP OVER SUBRTNS TO NEXT TEST.
1291
1292                                    ;SUBROUTINE FOR SLIDING BIT TESTS.
1293                                    ;
1294  005060  000000                    FILLR:  0                       ;PATTERN FOR FILLER.
1295  005062  000000                    SLIDR:  0                       ;BIT TO SLIDE.
1296  005064  013702  001420            SLIDE:  MOV     RAM,R2          ;FIRST LOC.
1297  005070  013703  001422                    MOV     RAMEND,R3
1298  005074  062703  000002                    ADD     #2,R3           ;LAST LOC +2.
1299  005100  013722  005060            1$:     MOV     FILLR,(R2)+     ;FILL WITH BACKGROUND
1300  005104  020203                    CMP     R2,R3           ;DONE YET ??
1301  005106  001374                    BNE     1$              ;NO.
1302
1303  005110  012737  005360  001110            MOV     #405,$LPERR     ;SET FOR SWR9 OPTION IN 'ERROR'
1304  005116  013702  001420                    MOV     RAM,R2          ;RESET 1ST LOC.
1305  005122  062703  177776                    ADD     #-2,R3          ;R3 = LAST LOC.
1306  005126  013701  005062                    MOV     SLIDR,R1        ;R1 = SLIDING BIT WORD
1307  005132  010112                    2$:     MOV     R1,(R2)         ;TEST 1ST (NEXT) LOC.
1308  005134  020112                    CMP     R1,(R2)         ;IS PATTERN RIGHT ??
1309  005136  001036                    BNE     10$             ;NO, PATTERN ERROR.
1310
1311  005140  020237  001420            11$:    CMP     R2,RAM          ;IS THIS THE 1ST LOC ??
1312  005144  001404                    BEQ     21$             ;YES, DONT CHECK LOC-2.
1313  005146  023762  005060  177776            CMP     FILLR,-2(R2)    ;IS LOC-2 UNDISTURBED ??
1314  005154  001037                    BNE     20$             ;NO, ERROR
1315
1316  005156  020203                    21$:    CMP     R2,R3           ;IS THIS LAST LOC ??
1317  005160  001424                    BEQ     6$              ;YES, YOURE ALL DONE.
1318
1319  005162  023762  005060  000002            CMP     FILLR,2(R2)     ;IS LOC+2 UNDISTURBED ??
1320  005170  001052                    BNE     30$             ;NO, ERROR.
1321
1322  005172  005737  005062            31$:    TST     SLIDR           ;CLEAR 'C', SLIDING 0 OR 1 ??
1323  005176  100403                    BMI     4$              ; 0, SKIP NEXT 3.
1324  005200  006101                    ROL     R1              ;SLIDE THE 1 LEFT 1.
```

```
1325  005202  103353                  BCC     2$              ;LOOP FOR 16 BITS.
1326  005204  000403                  BR      5$              ;OK, NOW SET FOR NEXT LOC.
1327  005206  000261          4$:     SEC                     ;SET 'C'.
1328  005210  006101                  ROL     R1              ;SLIDE THE 0 LEFT 1.
1329  005212  103747                  BCS     2$              ;LOOP FOR 16 BITS.
1330
1331  005214  013712  005060  5$:     MOV     FILLR,(R2)      ;RESET BACKGROUND PATTERN.
1332  005220  062702  000002          ADD     #2,R2           ;BUMP ADDRESS POINTER.
1333  005224  013701  005062          MOV     SLIDR,R1        ;RESET SLIDE WORD.
1334  005230  000740                  BR      2$              ;...AND DO IT ALL AGAIN.
1335
1336  005232  000207          6$:     RTS     PC              ;EXIT TO CALLER.
1337
1338                          ;;*****************************************************************
1339                          ;IF THE PATTERN IS STORED WRONG, OR IF AN ADJACENT LOCATION
1340                          ;WAS ALTERED, REPORT THAT FACT.
1341                          ;
1342  005234  010137  001124  10$:     MOV     R1,$GDDAT       ;GOOD DATA
1343  005240  010237  001122          MOV     R2,$BDADR       ;FAILING ADDRESS
1344  005244  011237  001126          MOV     (R2),$BDDAT     ;BAD DATA
1345  005250  104001                  ERROR   1               ;PRINT ERROR "ITEM 1".
1346  005252  000732                  BR      11$             ;CONTINUE AFTER ERROR. (SWR9=0)
1347                                                          ;...OR RETRY AT 40$ (SWR9=1)
1348
1349  005254  010237  001120  20$:     MOV     R2,$GDADR       ;BASE LOC
1350  005260  011237  001124          MOV     (R2),$GDDAT     ;...CONTENTS = SLIDE PATTERN
1351  005264  010237  001122          MOV     R2,$BDADR
1352  005270  062737  177776  001122  ADD     #-2,$BDADR      ;FAILING ADDR = LOC-2
1353  005276  016237  177776  001126  MOV     -2(R2),$BDDAT   ;...CONTENTS = BAD DATA
1354  005304  104004                  ERROR   4               ;PRINT ERROR "ITEM 4".
1355  005306  013762  005060  177776  MOV     FILLR,-2(R2)    ;REFRESH FAILED LOCATION
1356  005314  000720                  BR      21$             ;CONTINUE AFTER ERROR. (SWR9=0)
1357                                                          ;...OR RETRY AT 40$ (SWR9=1)
1358
1359  005316  010237  001120  30$:     MOV     R2,$GDADR       ;SAME AS ABOVE..
1360  005322  011237  001124          MOV     (R2),$GDDAT
1361  005326  010237  001122          MOV     R2,$BDADR
1362  005332  062737  000002  001122  ADD     #2,$BDADR       ;FAILING ADDR = LOC+2
1363  005340  016237  000002  001126  MOV     2(R2),$BDDAT
1364  005346  104005                  ERROR   5               ;PRINT ERROR "ITEM 5".
1365  005350  013762  005060  000002  MOV     FILLR,2(R2)     ;REFRESH FAILED LOCATION
1366  005356  000705                  BR      31$             ;CONTINUE AFTER ERROR. (SWR9=0)
1367                                                          ;...OR...
1368  005360  013777  005060  173534  40$:     MOV     FILLR,@$BDADR   ;REFRESH FAILED LOCATION...
1369  005366  000661                  BR      2$              ;...AND RETRY (SWR9=1)
1370                          ;;*****************************************************************
```

```
1371                                    ;;*****************************************************************
1372                                    ;*TEST 7        RAM -- ALTERNATE BIT PATTERN TEST.
1373                                    ;;*****************************************************************
1374  005370  000004                    TST7:   SCOPE
1375  005372  012737  000100  001160            MOV     #100,$TIMES     ;;DO 100 ITERATIONS
1376  005400  013702  001420                    MOV     RAM,R2          ;1ST LOC
1377  005404  013703  001422                    MOV     RAMEND,R3
1378  005410  062703  000002                    ADD     #2,R3           ;LAST LOC +2
1379  005414  012701  052525                    MOV     #52525,R1       ;INITIAL R1 = 052525
1380
1381  005420  010122                    1$:     MOV     R1,(R2)+        ;STORE A WORD.
1382  005422  005101                            COM     R1              ;COMPLIMENT IT
1383  005424  020203                            CMP     R2,R3           ;DONE ALL ??
1384  005426  001374                            BNE     1$              ;NO, STORE COMP IN NEXT LOC.
1385
1386  005430  012737  005532  001110            MOV     #30$,$LPERR     ;SET FOR SWR9 OPTION IN 'ERROR'
1387  005436  013702  001420                    MOV     RAM,R2          ;NOW COMPARE THEM.
1388  005442  020122                    2$:     CMP     R1,(R2)+        ;PATTERN RIGHT ??
1389  005444  001016                            BNE     3$              ;NO, PATTERN ERROR
1390
1391  005446  005101                    4$:     COM     R1              ;COMP PATTERN
1392  005450  020203                            CMP     R2,R3           ;COMPARED ALL ??
1393  005452  001373                            BNE     2$              ;NO, COMPARE NEXT
1394                                    ;
1395                                    ;AT THIS POINT 052525,125252,052525,125252, ETC HAS BEEN DONE.
1396                                    ;       NOW USE 125252,052525,125252, ETC.
1397                                    ;
1398  005454  013702  001420                    MOV     RAM,R2          ;RESET START LOC.
1399  005460  022701  052525                    CMP     #52525,R1       ;IS THIS 1ST PASS ??
1400  005464  001002                            BNE     5$              ;NO, YOU'RE ALL DONE
1401  005466  005101                            COM     R1              ;YES, SET UP FOR 2ND PASS.
1402  005470  000753                            BR      1$              ;...AND DO IT.
1403
1404  005472  013737  001106  001110  5$:       MOV     $LPADR,$LPERR   ;SET FOR SWR9 OPTION IN 'SCOPE'
1405  005500  000416                            BR      TST10           ;;...AND GO TO NEXT TEST.
1406
1407                                    ;;*****************************************************************
1408  005502  016237  177776  001126  3$:       MOV     -2(R2),$BDDAT   ;BAD DATA
1409  005510  010237  001120                    MOV     R2,$GDADR
1410  005514  062737  177776  001120            ADD     #-2,$GDADR      ;FAILING ADDRESS.
1411  005522  010137  001124                    MOV     R1,$GDDAT       ;GOOD DATA
1412  005526  104006                            ERROR   6               ;PRINT ERROR "ITEM 6".
1413  005530  000746                            BR      4$              ;CONTINUE AFTER ERROR. (SWR9=0)
1414                                                                    ;... OR ...
1415  005532  010142                    30$:    MOV     R1,-(R2)        ;REFRESH FAILED LOCATION...
1416  005534  000742                            BR      2$              ;...AND RETRY (SWR9=1)
1417                                    ;;*****************************************************************
```

```
1418                                    ;;*****************************************************************
1419                                    ;*TEST 10       RAM -- WRITE BYTE TEST.
1420                                    ;;*****************************************************************
1421  005536  000004                    TST10:  SCOPE
1422  005540  012737  000100  001160            MOV     #100,STIMES     ;;DO 100 ITERATIONS
1423  005546  005037  005640                    CLR     BKG             ;BACKGROUND PATTERN = 0.
1424  005552  005001                            CLR     R1              ;BYTE INDEX = 0.
1425  005554  012737  000377  005634            MOV     #377,PTRN       ;TEST PATTERN = 0,-1
1426  005562  004737  005642                    JSR     PC,BYTTST       ;FILL WITH ZEROS, WRITE LO...
1427                                                                    ;...BYTE -1, AND TEST WORD.
1428  005566  012701  000001                    MOV     #1,R1           ;BYTE INDEX = 1.
1429  005572  005137  005634                    COM     PTRN            ;TEST PATTERN = -1,0
1430  005576  004737  005642                    JSR     PC,BYTTST       ;FILL WITH ZEROS, WRITE HI...
1431                                                                    ;...BYTE -1, AND TEST WORD.
1432  005602  012737  177777  005640            MOV     #-1,BKG         ;BACKGROUND PATTERN = -1.
1433  005610  005001                            CLR     R1              ;BYTE INDEX = 0.
1434  005612  004737  005642                    JSR     PC,BYTTST       ;FILL WITH 1'S, WRITE LO...
1435                                                                    ;...BYTE 0, AND TEST WORD.
1436  005616  012701  000001                    MOV     #1,R1           ;BYTE INDEX = 1
1437  005622  005137  005634                    COM     PTRN            ;TEST PATTERN = 0,-1
1438  005626  004737  005642                    JSR     PC,BYTTST       ;FILL WITH 1'S, WRITE HI...
1439                                                                    ;...BYTE 0, AND TEST WORD.
1440  005632  000475                            BR      TST11           ;;DONE, GO TO NEXT TEST.
1441                                    ;
1442                                    ;SUBROUTINE FOR RAM BYTE TEST.
1443                                    ;
1444  005634  000000                    PTRN:   0                       ;0,-1 OR -1,0
1445  005636  000000                    WORDX:  0                       ;WORD ADDRESS.
1446  005640  000000                    BKG:    0                       ;BACKGROUND PATTERN
1447  005642  013702  001420            BYTTST: MOV     RAM,R2          ;INIT RAM ADDRESS
1448  005646  012703  000400                    MOV     #400,R3         ;...AND WORD COUNT.
1449  005652  013722  005640            1$:     MOV     BKG,(R2)+       ;FILL RAM WITH BACKGROUND.
1450  005656  005303                            DEC     R3
1451  005660  001374                            BNE     1$
1452
1453  005662  012737  006016  001110            MOV     #11$,$LPERR     ;SET FOR SWR9 OPTION IN 'ERROR'
1454  005670  013737  001420  005636            MOV     RAM,WORDX       ;SET WORD ADDRESS...
1455  005676  063701  005636                    ADD     WORDX,R1        ;...AND BYTE ADDRESS (R1).
1456  005702  012703  000400                    MOV     #400,R3
1457  005706  005737  005640            4$:     TST     BKG             ;BACKGROUND = 0 ??
1458  005712  001003                            BNE     2$              ;NO, WRITE A 0 BYTE.
1459  005714  112711  177777                    MOVB    #-1,(R1)        ;YES, WRITE A -1 BYTE...
1460  005720  000401                            BR      3$              ;...AND SKIP NEXT INSTR.
1461  005722  105011                    2$:     CLRB    (R1)            ;WRITE A 0 BYTE
1462
1463  005724  000240                    3$:     NOP
1464  005726  023777  005634  177702            CMP     PTRN,@WORDX     ;PATTERN RIGHT IN RAM ??
1465  005734  001013                            BNE     10$             ;NO, ERROR.
1466  005736  062737  000002  005636  5$:     ADD     #2,WORDX        ;YES, BUMP ADDRESS POINTERS.
1467  005744  062701  000002                    ADD     #2,R1
1468  005750  005303                            DEC     R3
1469  005752  001355                            BNE     4$              ;CONTINUE
1470  005754  013737  001106  001110            MOV     $LPADR,$LPERR   ;SET FOR SWR9 OPTION IN 'SCOPE'
1471  005762  000207                            RTS     PC              ;...AND EXIT.
```

```
1472
1473                                            ;;*****************************************************************
1474  005764  013737  005636  001120  10$:     MOV     WORDX,$GDADR    ;FAILING WORD ADDRESS
1475  007772  017737  177640  001126          MOV     @WORDX,$BDDAT   ;BAD DATA WORD.
1476  007000  010137  001122                  MOV     R1,$BDADR       ;BYTE WRITTEN ADDRESS
1477  006004  013737  005634  001124          MOV     PTRN,$GDDAT     ;GOOD DATA PATTERN
1478  006012  104007                          ERROR   7               ;PRINT ERROR "ITEM 7".
1479  006014  000750                          BR      5$              ;CONTINUE AFTER ERROR (SWR9=0)
1480                                                                  ;... OR ...
1481  006016  013777  005640  177612  11$:     MOV     BKG,@WORDX      ;REFRESH FAILED LOC...
1482  006024  000730                          BR      4$              ;...AND RETRY (SWR9=1)
1483                                            ;;*****************************************************************
```

```
1484                                    ;;*****************************************************************
1485                                    ;*TEST 11        PROM -- PROM TO RAM COPY TEST.
1486                                    ;;*****************************************************************
1487  006026  000004                    TST11:  SCOPE
1488  006030  012737  000100  001160            MOV     #100,$TIMES     ;;DO 100 ITERATIONS
1489  006036  013701  001424            MOV     PRM,R1          ;INIT PROM ADDRESS
1490  006042  013702  001420            MOV     RAM,R2          ;...AND RAM ADDRESS
1491  006046  012703  000400            MOV     #400,R3         ;... WORD COUNT
1492  006052  010304                    MOV     R3,R4           ;... DITTO
1493  006054  012737  006206  001110    MOV     #12$,$LPERR     ;SET FOR SWR9 OPTION IN 'ERROR'
1494
1495  006062  020102            1$:     CMP     R1,R2           ;IF RAM IS WITHIN PROM SPACE...
1496  006064  001002                    BNE     5$
1497  006066  062701  001000            ADD     #512.,R1        ;...SKIP 256 PROM LOCATIONS.
1498
1499  006072  012122            5$:     MOV     (R1)+,(R2)+     ;COPY PROM TO RAM
1500  006074  005303                    DEC     R3
1501  006076  001371                    BNE     1$
1502
1503  006100  162701  001000            SUB     #1000,R1        ;RESET PROM ADDRESS
1504  006104  013702  001420            MOV     RAM,R2          ;AND RAM ADDRESS
1505  006110  021112            2$:     CMP     (R1),(R2)       ;COMPARE PROM-RAM
1506  006112  001023                    BNE     10$             ;ERROR !!!
1507  006114  005304            11$:    DEC     R4              ;256 WORDS DONE ??
1508  006116  001403                    BEQ     3$              ;YES
1509  006120  005721                    TST     (R1)+           ;NO, BUMP POINTERS...
1510  006122  005722                    TST     (R2)+
1511  006124  000771                    BR      2$              ;...AND CONTINUE
1512
1513  006126  020137  001426    3$:     CMP     R1,PRMEND       ;END OF PROM ??
1514  006132  001407                    BEQ     4$              ;YES
1515  006134  005721                    TST     (R1)+           ;NO, BUMP PROM ADRESS
1516  006136  013702  001420            MOV     RAM,R2          ;INIT RAM ADDRESS...
1517  006142  012703  000400            MOV     #400,R3         ;...AND COUNTERS
1518  006146  010304                    MOV     R3,R4
1519  006150  000744                    BR      1$              ;AND DO THE NEXT CHUNK.
1520
1521  006152  013737  001106  001110    4$:     MOV     $LPADR,$LPERR   ;SET FOR SWR9 OPTION IN 'SCOPE'
1522  006160  000414                    BR      TST12           ;;...AND GO TO NEXT TEST.
1523
1524                                    ;;*****************************************************************
1525  006162  010137  001120            10$:    MOV     R1,$GDADR       ;PROM ADDRESS
1526  006166  011137  001124                    MOV     (R1),$GDDAT     ;PROM DATA
1527  006172  010237  001122                    MOV     R2,$BDADR       ;RAM ADDRESS
1528  006176  011237  001126                    MOV     (R2),$BDDAT     ;RAM DATA
1529  006202  104011                    ERROR   11              ;PRINT ERROR "ITEM 11".
1530  006204  000743                    BR      11$             ;CONTINUE AFTER ERROR (SWR9=0)
1531                                                            ;... OR ...
1532  006206  011112            12$:    MOV     (R1),(R2)       ;REFRESH FAILED RAM LOC...
1533  006210  000737                    BR      2$              ;...AND RETRY (SWR9=1)
1534                                    ;;*****************************************************************
```

```
                                ;;*****************************************************************
                                ;*TEST 12        PROM -- WRITE-TRAP TEST.
                                ;;*****************************************************************
006212  000004                  TST12:  SCOPE
006214  012737  000100  001160          MOV     #100,$TIMES     ;;DO 100 ITERATIONS
                                        .ENABL  LSB
006222  012737  006246  001110          MOV     #1$,$LPERR      ;SET FOR SWR9 OPTION IN 'ERROR'
006230  013701  001424                  MOV     PRM,R1          ;INIT PROM ADDRESS.

006234  020137  001420          5$:     CMP     R1,RAM          ;IF RAM IS WITHIN PROM SPACE...
006240  001002                          BNE     1$
006242  062701  001000                  ADD     #512.,R1        ;...SKIP 256 PROM LOCATIONS.

006246  011102                  1$:     MOV     (R1),R2         ;SAVE PROM WORD
006250  010211                          MOV     R2,(R1)         ;...AND TRY TO RE-WRITE IT.
                                                                ;IT SHOULD TRAP AND RETURN TO WTRTN

                                ;;*****************************************************************
                                ;NO-TRAP IS AN ERROR CONDITION.
                                ;
006252  010137  001122          WRT12:  MOV     R1,$BDADR       ;ERROR, FAILING ADDRESS
006256  010237  001124                  MOV     R2,$GDDAT       ;DATA WAS ...
006262  011137  001126                  MOV     (R1),$BDDAT     ;DATA IS ... (SHOULD BE SAME.)
006266  104012                          ERROR   12              ;PRINT ERROR "ITEM 12".
006270  000400                          BR      WTRTN           ;CONTINUE AFTER ERROR (SWR9=0)
                                                                ;...OR RETRY AT 1$ (SWR9=1)
                                ;;*****************************************************************
                                ;YOU'RE HERE IF THE WRITE ATTEMPT TRAPPED.
                                ;
006272  020137  001426          WTRTN:  CMP     R1,PRMEND       ;END OF PROM ??
006276  001402                          BEQ     4$              ;YES
006300  005721                          TST     (R1)+           ;NO, BUMP POINTER
006302  000754                          BR      5$              ;...AND DO THE NEXT ADDRESS.

006304  013737  001106  001110  4$:     MOV     $LPADR,$LPERR   ;SET FOR SWR9 OPTION IN 'SCOPE'
006312  000400                          BR      TST13           ;;DONE, NEXT TEST.
                                        .DSABL  LSB
```

```
                                        ;;*****************************************************************
                                        ;#TEST 13       PROM DATA TEST -- ENGINEERING USE ONLY.
                                        ;;*****************************************************************
006314  000004                          TST13:  SCOPE
006316  012737  000100  001160                  MOV     #100,$TIMES     ;;DO 100 ITERATIONS
                                        ;
                                        ;IF SPECIAL TEST PROM INSTALLED, DO THIS TEST.
                                        ;SKIP TO END-PASS OTHERWISE.
                                        ;
006324  005737  001432                  1$:     TST     PRMX            ;SPECIAL TEST PROM ??
006330  001010                                  BNE     XTEST           ;YES, EXECUTE THIS TEST.
006332  005037  001160                          CLR     $TIMES          ;NO, CLEAR ITER COUNT...
006336  000137  006540                          JMP     $EOP            ;NO, END PASS.
                                        ;
                                        ;FOR THIS TEST, A SPECIAL SET OF PROM CHIPS (8) MUST BE
                                        ;INSTALLED IN THE PROM BOARD.  THESE CHIPS (2/K) ARE BLASTED
                                        ;WITH THE FOLLOWING DATA PATTERN:
                                        ;
                                        ;       1ST K   010000 THRU 011777
                                        ;       2ND K   022000      023777
                                        ;       3RD K   044000      045777
                                        ;       4TH K   106000      107777
                                        ;
                                        ;NOTE THAT THE BIT PATTERN EMPLOYS TWO FIELDS AS FOLLOWS:
                                        ;       BITS 00-11 FORM A BINARY COUNTER, RANGE 0-7777 (4095.).
                                        ;       BITS 12-15 FORM A RING COUNTER, WHERE;
                                        ;                       BIT 12=1 IN 1ST K.
                                        ;                       BIT 13=1 IN 2ND K.
                                        ;                       BIT 14=1 IN 3RD K.
                                        ;                       BIT 15=1 IN 4TH K.
                                        ;
006342  010000                          K1:     BIT12                   ;BIT12=1 IN 1ST K.
006344  020000                          K2:     BIT13                   ;BIT13=1 IN 2ND K.
006346  040000                          K3:     BIT14                   ;BIT14=1 IN 3RD K.
006350  100000                          K4:     BIT15                   ;BIT15=1 IN 4TH K.
                                        
006352  000240                          XTEST:  NOP
006354  000240                                  NOP
006356  012737  006352  001106                  MOV     #XTEST,$LPADR   ;SET ITERATION ADDRESS
006364  012737  006420  001110                  MOV     #5$,$LPERR      ;SET FOR SWR9 OPTION IN 'ERROR'
006372  013701  006342                          MOV     K1,R1           ;INIT PATTERN FOR 1ST K.
006376  013702  001424                          MOV     PRM,R2          ;INIT ADDRESS.
                                        
006402  020237  001420                  6$:     CMP     R2,RAM          ;IF RAM IS WITHIN PROM SPACE...
006406  001004                                  BNE     5$
006410  062702  001000                          ADD     #512.,R2        ;...SKIP 256 PROM LOCATIONS.
006414  062701  000400                          ADD     #256.,R1        ;...ADJUST PATTERN ACCORDINGLY.
                                        
006420  020112                          5$:     CMP     R1,(R2)         ;COMPARE THIS(NEXT) LOCATION.
006422  001036                                  BNE     10$             ;ERROR
006424  005201                          11$:    INC     R1              ;BUMP PATTERN WORD...
006426  005722                                  TST     (R2)+           ;...AND ADDRESS.
                                        
006430  022701  012000                          CMP     #12000,R1       ;1ST K DONE ??
```

```
1626  006434  001412                          BEQ     1$              ;YES
1627  006436  022701  024000                  CMP     #24000,R1       ;2ND K DONE ??
1628  006442  001412                          BEQ     2$              ;YES
1629  006444  022701  046000                  CMP     #46000,R1       ;3RD K DONE ??
1630  006450  001412                          BEQ     3$              ;YES
1631  006452  022701  110000                  CMP     #110000,R1      ;4TH K DONE ??
1632  006456  001412                          BEQ     4$              ;YES
1633  006460  000757                          BR      5$              ;CONTINUE IN CURRENT SECTION.
1634
1635  006462  063701  006342          1$:     ADD     K1,R1           ;INIT PATTERN FOR 2ND K.
1636  006466  000745                          BR      6$
1637  006470  063701  006344          2$:     ADD     K2,R1           ;INIT PATTERN FOR 3RD K.
1638  006474  000742                          BR      6$
1639  006476  063701  006346          3$:     ADD     K3,R1           ;INIT PATTERN FOR 4TH K.
1640  006502  000737                          BR      6$
1641
1642  006504  000240                  4$:     NOP
1643  006506  000240                          NOP
1644  006510  013737  001106  001110          MOV     $LPADR,$LPERR   ;SET FOR SWR9 OPTION IN 'SCOPE'
1645  006516  000410                          BR      $EOP            ;;DONE, END-OF-PASS
1646
1647                                  ;;**************************************************************
1648  006520  010237  001122          10$:    MOV     R2,$BDADR       ;FAILING ADDRESS
1649  006524  010137  001124                  MOV     R1,$GDDAT       ;GOOD DATA
1650  006530  011237  001126                  MOV     (R2),$BDDAT     ;BAD DATA
1651  006534  104010                          ERROR   10              ;PRINT ERROR "ITEM 10".
1652  006536  000732                          BR      11$             ;...AND GO ON (SWR9=0).
1653                                                                  ;...OR RETRY AT 5$ (SWR9=1)
1654                                  ;;**************************************************************
```

```
                                .SBTTL  END OF PASS ROUTINE
                                ;
                                ;;*********************************************************************
                                ;*INCREMENT THE PASS NUMBER ($PASS)
                                ;*TYPE "END PASS #XXXXX" (WHERE XXXXX IS A DECIMAL NUMBER)
                                ;*IF THERES A MONITOR GO TO IT
                                ;*IF THERE ISN'T JUMP TO AGAIN
                                ;
006540                          SEOP:
                                        SCOPE
000004
                                        CLR     $TSTNM          ;;ZERO THE TEST NUMBER
001102                                  CLR     $ITERS          ;;ZERO THE NUMBER OF ITERATIONS
001160                                  INC     $               ;;INCREMENT THE PASS NUMBER
001202                                  BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
100000  001202                          DEC     (PC)+           ;;LOOP?
001202                          SEOPCT: .WORD   1
                                        BGT     $DOAGN          ;;YES
                                        MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
000001                          SENDCT: .WORD   1
                                        $EOPCT
                                        TYPE    ,$ENDMG         ;;TYPE "END PASS #"
001202                                  MOV     $PASS,-(SP)     ;;SAVE $PASS FOR TYPEOUT
                                        TYPOS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
                                        TYPE    ,$ENULL         ;;TYPE A NULL CHARACTER
000042                          $GET42: MOV     @#42,R0         ;;GET MONITOR ADDRESS
                                        BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
                                        RESET                   ;;CLEAR THE WORLD
                                SENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
                                        NOP                     ;;SAVE ROOM
                                        NOP                     ;;FOR
                                        NOP                     ;;ACT11
                                $DOAGN:
                                        JMP     @(PC)+          ;;RETURN
002624                          $RTNAD: .WORD   AGAIN
      377       377     000     $ENULL: .BYTE   -1,-1,0         ;;NULL CHARACTER STRING
      015   042412  042116      $ENDMG: .ASCIZ  <15><12>/END PASS #/
050040  051501  020123
000043
```

```
                                .SBTTL  COMMON SUBROUTINES AND TABLES
                                ;;*****************************************************************
                                ;THIS ROUTINE WILL TEST FOR A NEXM IN THE RANGE (R1) THRU (R2).
                                ;
                                ;CALL:  MOV ADR1,R1
                                ;       MOV ADR2,R2
                                ;       JSR PC,NXM
                                ;       RETURN IF BUSERR TRAP
                                ;       RETURN IF NO-TRAP
                                ;
001662  005711                  NXM:    TST     (R1)            ;TEST THE ADDRESSES...
001664  005712                          TST     (R2)            ;...IF EITHER TRAPS, RETURN TO NXM1.
001666  062716  000002                  ADD     #2,(SP)         ;IF NOT, BUMP RETURN PC...
006672  000207                          RTS     PC              ;...AND RETURN TO CALL+6.
006674  000207                  NXM1:   RTS     PC              ;ONE OR THE OTHER TRAPPED...
                                                                ;...RETURN TO CALL+4.
                                ;
                                ;;*****************************************************************
                                ;BUS TIME-OUT PROCESSER.  BUS ERRORS ARE EXPECTED DURING
                                ;INITIALIZATION, AND IN TESTS 1 AND 12.
                                ;IF ANY OTHER, REPORT BUS-ERROR, AND HALT.
                                ;
006676  022716  002740          TIMOUT: CMP     #RD1,(SP)       ;TST1 READ TRAP ??
006702  001003                          BNE     1$              ;NO
006704  012716  002774                  MOV     #RTN1,(SP)      ;YES, ADJUST RETURN PC
006710  000002                          RTI                     ;AND RETURN THERE.
                                ;
006712  022716  002744          1$:     CMP     #WRT1,(SP)      ;TST1 WRITE TRAP ??
006716  001003                          BNE     2$              ;NO
006720  012716  003024                  MOV     #RTN2,(SP)      ;YES...
006724  000002                          RTI                     ;...RETURN THERE
                                ;
006726  022716  006252          2$:     CMP     #WRT12,(SP)     ;TST12 WRITE TRAP ??
006732  001003                          BNE     3$              ;NO
006734  012716  006272                  MOV     #WTRTN,(SP)     ;YES...
006740  000002                          RTI                     ;...RETURN THERE.
                                ;
006742  022716  006664          3$:     CMP     #NXM+2,(SP)     ;NEXM ROUTINE TRAP 1 ??
006746  001403                          BEQ     4$              ;YES
006750  022716  006666                  CMP     #NXM+4,(SP)     ;NO, NEXM TRAP 2 ??
006754  001003                          BNE     5$              ;NO
006756  012716  006674          4$:     MOV     #NXM1,(SP)      ;RETURN TO NEXM ROUTINE.
006762  000002                          RTI
                                ;
006764                          5$:
006764  012637  001122                  MOV     (SP)+,$BOADR    ;;POP STACK INTO $BOADR
006770  012637  001126                  MOV     (SP)+,$BOOAT    ;;POP STACK INTO $BOOAT
006774  012737  007004  001110          MOV     #STOP,@LPERR    ;OVERRIDE SWR9 OPTION.
007002  104013                          ERROR   13              ;REPORT BUS ERROR...
                                                                ;...$BOADR = TRAP PC
                                                                ;...$BOOAT = TRAP PS
007004  000000          STOP:   HALT                            ;STOP...
007006  000137  001436                  JMP     START           ;...RESTART ON "CONTINUE".
```

```
                                ;*****************************************************************
                                ;MEMORY MAP FOR TEST 1.
                                ;THE MAP CONSISTS OF 2 WORDS FOR EACH 4K BANK THRU 32K.
                                ;
                                ;WORD 1 IS THE READ MAP WHERE EACH BIT REPRESENTS 1 256 WORD
                                ;SEGMENT OF THE BANK. EACH BIT IS SET 0 IF THAT SEGMENT IS
                                ;"NEXM", SET 1 OTHERWISE.
                                ;
                                ;WORD 2 IS THE WRITE MAP FOR THE SAME BANK.  EACH BIT IS
                                ;SET 0 IF THAT SEGMENT IS "READ-ONLY" OR "NEXM", SET 1 OTHERWISE.
                                ;
                                ;BIT# TO ADDRESS CORRESPONDENCE IS AS FOLLOWS:
                                ;
                                ;       BIT #   RANGE
                                ;       15      0 - 776
                                ;       14      1000 - 1776
                                ;       13      2000 - 2776
                                ;       12      3000 - 3776
                                ;       11      4000 - 4776
                                ;       10      5000 - 5776
                                ;       09      6000 - 6776
                                ;       08      7000 - 7776
                                ;       07      10000 - 10776
                                ;       06      11000 - 11776
                                ;       05      12000 - 12776
                                ;       04      13000 - 13776
                                ;       03      14000 - 14776
                                ;       02      15000 - 15776
                                ;       01      16000 - 16776
                                ;       00      17000 - 17776
                                ;
007012  007024          RMAP:   .WORD   MEMMAP          ;POINTS TO 1ST WORD IN PAIR.
007014  007026          WMAP:   .WORD   MEMMAP+2        ;POINTS TO 2ND WORD IN PAIR.
007016  000020          MAPK:   .WORD   16.             ;BIT (SEGMENT) COUNTER.
007020  000000          LOC:    .WORD   0               ;POINTS TO LOC TO TEST.
007022  000     000     MAPSW:  .BYTE   0,0             ;LO BYTE IS TYPE/INHIB SWITCH
                                                        ;HI BYTE IS COPY/COMPARE SWITCH
007024  000020          MEMMAP: .BLKW   16.             ;THIS IS THE WORKING (SCRATCH) MAP.
                                                        ;2 WORDS/4K BANK AS DESCRIBED ABOVE.
007064  000020          SAVMAP: .BLKW   16.             ;INITIAL COPY OF "MEMMAP" IS SAVED
                                                        ;HERE AS THE REFERENCE MAP.
                                ;
                                ;;*****************************************************************
                                ;SUBROUTINE TO DUMP A 32K MAP AND HALT.  START AT LOC 220.
                                ;
001456                  STARTM=START+20
007124  012737  177777  001434  MDUMP:  MOV     #-1,MAINT       ;SET MAINT FLAG.
007132  012706  001100          MOV     #STACK,SP       ;INIT THE STACK
007136  004737  001456          JSR     PC,STARTM       ;INIT VECTORS, AND RETURN.
007142  005037  007022          CLR     MAPSW           ;ENABLE MAP OUTPUT.
007146  004737  002670          JSR     PC,TST1A        ;...DO IT...
007152  005037  001434          CLR     MAINT           ;...RETURN...
007156  000000                  HALT                    ;...AND HALT.
007160  000137  007124          JMP     MDUMP
```

```
                                        ;;*****************************************************************
                                        ;THIS ROUTINE WILL OUTPUT ANY GIVEN MEM RANGE TO THE
                                        ;CONSOLE TTY.
                                        ;
                                        ;START AT LOC 210 TO DUMP OCTAL,
                                        ;     AND LOC 214 TO DUMP BINARY.
                                        ;
                                        ;OPERATOR IS REQUIRED TO INPUT FIRST ADDRESS,
                                        ;AND NUMBER OF WORDS TO DUMP.
                                        ;
                                        ;AN ODD ADDRESS INPUT WILL BE DECREMENTED BY 1, BEFORE USE.
                                        ;WORD COUNT INPUT MUST BE NON-ZERO.
                                        ;RESTART DUMPER AT END.
                                        ;
007164  000000                  MODE:   0                       ; 0 = OCTAL, -1 = BINARY
007166  000U00                  ADDRS:  0                       ;ADDRESS OF DUMP RANGE
007170  000000                  WORDS:  0                       ;WORD COUNT
007172  000000                  LINE:   0                       ;NUMBER OF WORDS/LINE

007174  012737  177777  001434  ODUMP:  MOV     #-1,MAINT       ;SET MAINT FLAG
007202  012706  001100                  MOV     #STACK,SP       ;INIT THE STACK
007206  004737  001456                  JSR     PC,STARTM       ;INIT VECTORS, AND RETURN
007212  005037  001434                  CLR     MAINT
007216  104401  007224                  TYPE    ,65$            ;;TYPE ASCIZ STRING
007222  000417                          BR      64$             ;;GET OVER THE ASCIZ
                                ;;65$:  .ASCIZ  <CRLF><CRLF>/OCTAL MEMORY DUMP ROUTINE./<CRLF>
007262                          64$:
007262  005037  007164                  CLR     MODE            ; 0 = OCTAL DUMP
007266  012737  000004  007172          MOV     #4,LINE         ;WITH 4 WORDS PER LINE.
007274  000442                          BR      DSTRT

007276  012737  177777  001434  BDUMP:  MOV     #-1,MAINT       ;SET MAINT FLAG
007304  012706  001100                  MOV     #STACK,SP       ;INIT THE STACK
007310  004737  001456                  JSR     PC,STARTM       ;INIT VECTORS, AND RETURN
007314  005037  001434                  CLR     MAINT
007320  104401  007326                  TYPE    ,65$            ;;TYPE ASCIZ STRING
007324  000420                          BR      64$             ;;GET OVER THE ASCIZ
                                ;;65$:  .ASCIZ  <CRLF><CRLF>/BINARY MEMORY DUMP ROUTINE./<CRLF>
007366                          64$:
007366  012737  177777  007164          MOV     #-1,MODE        ;-1 = BINARY DUMP
007374  012737  000002  007172          MOV     #2,LINE         ;WITH 2 WORDS PER LINE.

007402                          DSTRT:
007402  104401  007410                  TYPE    ,65$            ;;TYPE ASCIZ STRING
007406  000411                          BR      64$             ;;GET OVER THE ASCIZ
                                ;;65$:  .ASCIZ  <CRLF>/FIRST ADDRESS = /
007432                          64$:
007432  104413                          RDOCT                   ;RECEIVE HIS RESPONSE.
007434  012637  007166                  MOV     (SP)+,ADDRS     ;;POP STACK INTO ADDRS
007440  042737  000001  007166          BIC     #BIT0,ADDRS     ;INSURE ITS AN EVEN ADDRESS.

007446                          2$:
007446  104401  007454                  TYPE    ,67$            ;;TYPE ASCIZ STRING
007452  000414                          BR      66$             ;;GET OVER THE ASCIZ
```

```
1854                                    ;;67$:  .ASCIZ  <CRLF>/WORD COUNT (OCTAL) = /
1855  007504                            66$:
1856  007504  104413                            RDOCT                   ;GET THE NUMBER
1857  007506  012637  007170                    MOV     (SP)+,WORDS     ;;POP STACK INTO WORDS
1858  007512  001755                            BEQ     2$              ;CANT USE 0, ASK FOR ANOTHER.
1859
1860  007514  013701  007166                    MOV     ADDRS,R1        ;FIRST ADDRESS IN R1.
1861  007520  013702  007170                    MOV     WORDS,R2        ;WORD COUNT...
1862  007524  006302                            ASL     R2              ;...TIMES 2 = BYTE COUNT.
1863  007526  062702  177776                    ADD     #-2,R2          ;ADJUST
1864  007532  060102                            ADD     R1,R2           ;PLUS (R1) = LAST ADDR IN R2.
1865  007534  004737  006662                    JSR     PC,NXM          ;SEE IF THE RANGE IS VALID.
1866  007540  000401                            BR      10$             ;ITS NOT, RANGE IS NEXM.
1867  007542  000403                            BR      DMPGO           ;IT IS, START DUMPING.
1868
1869  007544  104401  013412          10$:     TYPE    ,M3             ;INVALID DUMP RANGE.
1870  007550  000714                            BR      DSTRT           ;TRY AGAIN.
1871
1872  007552  104401  001171          DMPGO:   TYPE    ,$CRLF
1873  007556  013746  007166                    MOV     ADDRS,-(SP)     ;;PUSH ADDRS ON STACK
1874  007562  104403                            TYPOS                   ;TYPE THE ADDRESS.
1875  007564     006                            .BYTE   6
1876  007565     000                            .BYTE   0               ;SUPPRESS LEAD ZEROS.
1877  007566  013701  007172                    MOV     LINE,R1         ;SET WORDS/LINE COUNT
1878
1879  007572  104401  007650          3$:      TYPE    ,SPACE3         ;3 SPACES BETWEEN WORDS
1880  007576  017746  177364                    MOV     @ADDRS,-(SP)    ;;PUSH @ADDRS ON STACK
1881  007602  005737  007164                    TST     MODE
1882  007606  001402                            BEQ     1$
1883  007610  104406                            TYPBN                   ;TYPE BINARY DATA.
1884  007612  000401                            BR      2$
1885
1886  007614  104402                    1$:     TYPOC                   ;TYPE OCTAL DATA.
1887  007616  062737  000002  007166  2$:      ADD     #2,ADDRS        ;BUMP ADDRESS POINTER.
1888  007624  005337  007170                    DEC     WORDS           ;AND WORD COUNT
1889  007630  001403                            BEQ     4$              ;QUIT WHEN WC = 0.
1890
1891  007632  005301                            DEC     R1              ;BUMP WORDS/LINE COUNT
1892  007634  001746                            BEQ     DMPGO           ;START A NEW LINE
1893  007636  000755                            BR      3$              ;NEXT WORD, THIS LINE
1894
1895  007640  104401  001171          4$:      TYPE    ,$CRLF
1896  007644  000240                            NOP
1897  007646  000655                            BR      DSTRT           ;RESTART ON 'CONT'
1898
1899  007650     040     040     040  SPACE3:  .BYTE   40,40,40,0      ;3 SPACES
1900  007653     000
```

```
1901                                    .SBTTL  SYSMAC UTILITIES FOLLOW:
1902                                    .SBTTL  SCOPE HANDLER ROUTINE
1903
1904                                    ;;**************************************************************
1905                                    ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
1906                                    ;*AND LOAD THE TEST NUMBER($STSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
1907                                    ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
1908                                    ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
1909                                    ;*SW14=1         LOOP ON TEST
1910                                    ;*SW11=1         INHIBIT ITERATIONS
1911                                    ;*SW09=1         LOOP ON ERROR
1912                                    ;*SW08=1         LOOP ON TEST IN SWR<7:0>
1913                                    ;*CALL
1914                                    ;*      SCOPE           ;;SCOPE=IOT
1915
1916  007654                            $SCOPE:
1917  007654  104410                            CKSWR                   ;;TEST  FOR CHANGE IN SOFT-SWR
1918  007656  032777  040000  171254    1$:     BIT     #BIT14,@SWR     ;;LOOP ON PRESENT TEST?
1919  007664  001131                            BNE     $OVER           ;;YES IF SW14=1
1920                                    ;#####START OF CODE FOR THE XOR TESTER#####
1921  007666  000416                    $XTSTR: BR      6$              ;;IF RUNNING ON THE "XOR" TESTER CHANGE
1922                                                                    ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
1923  007670  013746  000004                   MOV     @#ERRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
1924  007674  012737  007714  000004          MOV     #5$,@#ERRVEC    ;;SET FOR TIMEOUT
1925  007702  005737  177060                   TST     @#177060        ;;TIME OUT ON XOR?
1926  007706  012637  000004                   MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
1927  007712  000500                           BR      $SVLAD          ;;GO TO THE NEXT TEST
1928  007714  022626                    5$:     CMP     (SP)+,(SP)+     ;;CLEAR THE STACK AFTER A TIME OUT
1929  007716  012637  000004                   MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
1930  007722  000440                           BR      7$              ;;LOOP ON THE PRESENT TEST
1931  007724                            6$:;#####END OF CODE FOR THE XOR TESTER#####
1932  007724  032777  000400  171206        BIT     #BIT08,@SWR     ;;LOOP ON SPEC. TEST?
1933  007732  001421                            BEQ     2$              ;;BR IF NO
1934  007734  005046                            CLR     -(SP)           ;;CLEAR A TEMP. LOCATION
1935  007736  117716  171176                   MOVB    @SWR,(SP)       ;;PICKUP THE DESIRED TEST NUMBER
1936  007742  001414                            BEQ     8$              ;;BRANCH IF BAD TEST NUMBER IN SWR
1937  007744  022716  000013                   CMP     #13,(SP)        ;;CHECK THE NUMBER IN THE SWR
1938  007750  002411                            BLT     8$              ;;BRANCH IF TEST NUMBER IS OUT OF RANGE
1939  007752  011637  001102                   MOV     (SP),$TSTNM     ;;UPDATE THE TEST NUMBER
1940  007756  005316                            DEC     (SP)            ;;BACKUP BY ONE
1941  007760  006316                            ASL     (SP)            ;;SCALE THE TEST NUMBER AS AN INDEX
1942  007762  062716  010166                   ADD     #$SWR0BTBL,(SP) ;;FORM THE ADDRESS OF TEST POINTER
1943  007766  013637  001106                   MOV     @(SP)+,$LPADR   ;;SET LOOP ADDRESS TO DESIRED TEST
1944  007772  000466                           BR      $OVER           ;;GO LOOP ON THE TEST
1945  007774  005726                    8$:     TST     (SP)+           ;;CLEAN THE BAD TEST NUMBER OFF OF THE STACK
1946  007776  105737  001103            2$:     TSTB    $ERFLG          ;;HAS AN ERROR OCCURRED?
1947  010002  001421                            BEQ     3$              ;;BR IF NO
1948  010004  123737  001115  001103          CMPB    $ERMAX,$ERFLG   ;;MAX. ERRORS FOR THIS TEST OCCURRED?
1949  010012  101015                            BHI     3$              ;;BR IF NO
1950  010014  032777  001000  171116          BIT     #BIT09,@SWR     ;;LOOP ON ERROR?
1951  010022  001404                            BEQ     4$              ;;BR IF NO
1952  010024  013737  001110  001106    7$:     MOV     $LPERR,$LPADR   ;;SET LOOP ADDRESS TO LAST SCOPE
1953  010032  000446                            BR      $OVER
1954  010034  105037  001103            4$:     CLRB    $ERFLG          ;;ZERO THE ERROR FLAG
```

```
 1955  010040  005037  001160                    CLR     $TIMES          ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
 1956  010044  000415                            BR      1$              ;;ESCAPE TO THE NEXT TEST
 1957  010046  032777  004000  171064  3$:       BIT     #BIT11,@SWR     ;;INHIBIT ITERATIONS?
 1958  010054  001011                            BNE     1$              ;;BR IF YES
 1959  010056  005737  001202                    TST     $PASS           ;;IF FIRST PASS OF PROGRAM
 1960  010062  001406                            BEQ     1$              ;;    INHIBIT ITERATIONS
 1961  010064  005237  001104                    INC     $ICNT           ;;INCREMENT ITERATION COUNT
 1962  010070  023737  001160  001104            CMP     $TIMES,$ICNT    ;;CHECK THE NUMBER OF ITERATIONS MADE
 1963  010076  002024                            BGE     $OVER           ;;BR IF MORE ITERATION REQUIRED
 1964  010100  012737  000001  001104  1$:       MOV     #1,$ICNT        ;;REINITIALIZE THE ITERATION COUNTER
 1965  010106  013737  010164  001160            MOV     $MXCNT,$TIMES   ;;SET NUMBER OF ITERATIONS TO DO
 1966  010114  105237  001102          $SVLAD:   INCB    $TSTNM          ;;COUNT TEST NUMBERS
 1967  010120  113737  001102  001200            MOVB    $TSTNM,$TESTN   ;;SET TEST NUMBER IN APT MAILBOX
 1968  010126  01.637  001106                    MOV     (SP),$LPADR     ;;SAVE SCOPE LOOP ADDRESS
 1969  010132  011637  001110                    MOV     (SP),$LPERR     ;;SAVE ERROR LOOP ADDRESS
 1970  010136  005037  001162                    CLR     $ESCAPE         ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
 1971  010142  112737  000001  001115            MOVB    #1,$ERMAX       ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
 1972  010150  013777  001102  170764  $OVER:    MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
 1973  010156  013716  001106                    MOV     $LPADR,(SP)     ;;FUDGE RETURN ADDRESS
 1974  010162  000002                            RTI                     ;;FIXES PS
 1975  010164  003720                  $MXCNT:   2000.                   ;;MAX. NUMBER OF ITERATIONS
 1976  010166                           $SWO8TBL:
 1977  010166  002632                            .WORD   TST1+2          ;;STARTING ADDRESS OF TEST 1
 1978  010170  003364                            .WORD   TST2+2          ;;STARTING ADDRESS OF TEST 2
 1979  010172  003546                            .WORD   TST3+2          ;;STARTING ADDRESS OF TEST 3
 1980  010174  003662                            .WORD   TST4+2          ;;STARTING ADDRESS OF TEST 4
 1981  010176  004766                            .WORD   TST5+2          ;;STARTING ADDRESS OF TEST 5
 1982  010200  005024                            .WORD   TST6+2          ;;STARTING ADDRESS OF TEST 6
 1983  010202  005372                            .WORD   TST7+2          ;;STARTING ADDRESS OF TEST 7
 1984  010204  005540                            .WORD   TST10+2         ;;STARTING ADDRESS OF TEST 10
 1985  010206  006030                            .WORD   TST11+2         ;;STARTING ADDRESS OF TEST 11
 1986  010210  006214                            .WORD   TST12+2         ;;STARTING ADDRESS OF TEST 12
 1987  010212  006316                            .WORD   TST13+2         ;;STARTING ADDRESS OF TEST 13
```

```
                                                .SBTTL  ERROR HANDLER ROUTINE
                                                ;;**********************************************************************
                                                ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
                                                ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
                                                ;*AND GO TO $ERRTYP ON ERROR
                                                ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
                                                ;*SW15=1         HALT ON ERROR
                                                ;*SW13=1         INHIBIT ERROR TYPEOUTS
                                                ;*SW10=1         BELL ON ERROR
                                                ;*SW09=1         LOOP ON ERROR
                                                ;*CALL
                                                ;*      ERROR   N       ;;ERROR=EMT AND N=ERROR ITEM NUMBER

010214                                          $ERROR:
010214  104410                                          CKSWR                           ;;TEST FOR CHANGE IN SOFT-SWR
010216  105237  001103                          7$:     INCB    $ERFLG                  ;;SET THE ERROR FLAG
010222  001775                                          BEQ     7$                      ;;DON'T LET THE FLAG GO TO ZERO
010224  013777  001102  170710                          MOV     $TSTNM,@DISPLAY         ;;DISPLAY TEST NUMBER AND ERROR FLAG
010232  032777  002000  170700                          BIT     #BIT10,@SWR             ;;BELL ON ERROR?
010240  001402                                          BEQ     1$                      ;;NO - SKIP
010242  104401  001164                                  TYPE    ,$BELL                  ;;RING BELL
010246  005237  001112                          1$:     INC     $ERTTL                  ;;COUNT THE NUMBER OF ERRORS
010252  011637  001116                                  MOV     (SP),$ERRPC             ;;GET ADDRESS OF ERROR INSTRUCTION
010256  162737  000002  001116                          SUB     #2,$ERRPC               ;;
010264  117737  170626  001114                          MOVB    @$ERRPC,$ITEMB          ;;STRIP AND SAVE THE ERROR ITEM CODE
010272  032777  020000  170640                          BIT     #BIT13,@SWR             ;;SKIP TYPEOUT IF SET
010300  001004                                          BNE     20$                     ;;SKIP TYPEOUTS
010302  004737  010414                                  JSR     PC,$ERRTYP              ;;GO TO USER ERROR ROUTINE
010306  104401  001171                                  TYPE    ,$CRLF
010312                                          20$:
010312  122737  000001  001214                          CMPB    #APTENV,$ENV            ;;RUNNING IN APT MODE
010320  001007                                          BNE     2$                      ;;NO,SKIP APT ERROR REPORT
010322  113737  001114  010334                          MOVB    $ITEMB,21$              ;;SET ITEM NUMBER AS ERROR NUMBER
010330  004737  012670                                  JSR     PC,$ATY4                ;;REPORT FATAL ERROR TO APT
010334     000                                  21$:    .BYTE   0
010335     000                                          .BYTE   0
010336  000777                                  22$:    BR      22$                     ;;APT ERROR LOOP
010340  005777  170574                          2$:     TST     @SWR                    ;;HALT ON ERROR
010344  100002                                          BPL     3$                      ;;SKIP IF CONTINUE
010346  000000                                          HALT                            ;;HALT ON ERROR!
010350  104410                                          CKSWR                           ;;TEST FOR CHANGE IN SOFT-SWR
010352  032777  001000  170560                  3$:     BIT     #BIT09,@SWR             ;;LOOP ON ERROR SWITCH SET?
010360  001402                                          BEQ     4$                      ;;BR IF NO
010362  013716  001110                                  MOV     $LPERR,(SP)             ;;FUDGE RETURN FOR LOOPING
010366  005737  001162                          4$:     TST     $ESCAPE                 ;;CHECK FOR AN ESCAPE ADDRESS
010372  001402                                          BEQ     5$                      ;;BR IF NONE
010374  013716  001162                                  MOV     $ESCAPE,(SP)            ;;FUDGE RETURN ADDRESS FOR ESCAPE
010400                                          5$:
010400  022737  006626  000042                          CMP     #$ENDAD,@#42            ;;ACT-11 AUTO-ACCEPT?
010406  001001                                          BNE     6$                      ;;BRANCH IF NO
010410  000000                                          HALT                            ;;YES
010412                                          6$:
010412  000002                                          RTI                             ;;RETURN
```

```
                                .SBTTL  ERROR MESSAGE TYPEOUT ROUTINE

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

010414                          $ERRTYP:
010414  104401  001171                  TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
010420  010046                          MOV     R0,-(SP)        ;;SAVE R0
010422  005000                          CLR     R0              ;;PICKUP THE ITEM INDEX
010424  153700  001114                  BISB    @#$ITEMB,R0
010430  001004                          BNE     1$              ;;IF ITEM NUMBER IS ZERO, JUST
                                                                ;;TYPE THE PC OF THE ERROR
010432  013746  001116                  MOV     $ERRPC,-(SP)    ;;SAVE $ERRPC FOR TYPEOUT
                                                                ;;ERROR ADDRESS
010436  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
010440  000426                          BR      6$              ;;GET OUT
010442  005300                  1$:     DEC     R0              ;;ADJUST THE INDEX SO THAT IT WILL
010444  006300                          ASL     R0              ;;      WORK FOR THE ERROR TABLE
010446  006300                          ASL     R0
010450  006300                          ASL     R0
010452  062700  001260                  ADD     #$ERRTB,R0      ;;FORM TABLE POINTER
010456  012037  010466                  MOV     (R0)+,2$        ;;PICKUP "ERROR MESSAGE" POINTER
010462  001404                          BEQ     3$              ;;SKIP TYPEOUT IF NO POINTER
010464  104401                          TYPE                    ;;TYPE THE "ERROR MESSAGE"
010466  000000                  2$:     .WORD   0               ;;"ERROR MESSAGE" POINTER GOES HERE
010470  104401  001171                  TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
010474  012037  010504          3$:     MOV     (R0)+,4$        ;;PICKUP "DATA HEADER" POINTER
010500  001404                          BEQ     5$              ;;SKIP TYPEOUT IF 0
010502  104401                          TYPE                    ;;TYPE THE "DATA HEADER"
010504  000000                  4$:     .WORD   0               ;;"DATA HEADER" POINTER GOES HERE
010506  104401  001171                  TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
010512  011000                  5$:     MOV     (R0),R0         ;;PICKUP "DATA TABLE" POINTER
010514  001004                          BNE     7$              ;;GO TYPE THE DATA
010516  012600                  6$:     MOV     (SP)+,R0        ;;RESTORE R0
010520  104401  001171                  TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
010524  000207                          RTS     PC              ;;RETURN
010526                          7$:
010526  013046                          MOV     @(R0)+,-(SP)    ;;SAVE @(R0)+ FOR TYPEOUT
010530  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
010532  005710                          TST     (R0)            ;;IS THERE ANOTHER NUMBER?
010534  001770                          BEQ     6$              ;;BR IF NO
010536  104401  010544                  TYPE    ,8$             ;;TYPE TWO(2) SPACES
010542  000771                          BR      7$              ;;LOOP
010544  020040      000         8$:     .ASCIZ  /  /            ;;TWO(2) SPACES
010550                                  .EVEN
```

```
                                        .SBTTL  TYPE ROUTINE
                                        ;;*****************************************************************
                                        ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
                                        ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
                                        ;*NOTE1:         $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
                                        ;*NOTE2:         $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
                                        ;*NOTE3:         $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
                                        ;*
                                        ;*CALL:
                                        ;*1) USING A TRAP INSTRUCTION
                                        ;*       TYPE    ,MESADR         ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
                                        ;*OR
                                        ;*       TYPE
                                        ;*       MESADR
                                        ;*
010550  105737  001157          $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
010554  100002                          BPL     1$              ;;BR IF YES
010556  000000                          HALT                    ;;HALT HERE IF NO TERMINAL
010560  000430                          BR      3$              ;;LEAVE
010562  010046                  1$:     MOV     R0,-(SP)        ;;SAVE R0
010564  017600  000002                  MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
010570  122737  000001  001214          CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
010576  001011                          BNE     62$             ;;NO,GO CHECK FOR APT CONSOLE
010600  132737  000100  001215          BITB    #APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
010606  001405                          BEQ     62$             ;;NO,GO CHECK FOR CONSOLE
010610  010037  010620                  MOV     R0,61$          ;;SETUP MESSAGE ADDRESS FOR APT
010614  004737  012660                  JSR     PC,$ATY3        ;;SPOOL MESSAGE TO APT
010620  000000                  61$:    .WORD   0               ;;MESSAGE ADDRESS
010622  132737  000040  001215  62$:    BITB    #APTCSUP,$ENVM  ;;APT CONSOLE SUPPRESSED
010630  001003                          BNE     60$             ;;YES,SKIP TYPE OUT
010632  112046                  2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
010634  001005                          BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
010636  005726                          TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
010640  012600                  60$:    MOV     (SP)+,R0        ;;RESTORE R0
010642  062716  000002          3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
010646  000002                          RTI                     ;;RETURN
010650  122716  000011          4$:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
010654  001430                          BEQ     8$
010656  122716  000200                  CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
010662  001006                          BNE     5$
010664  005726                          TST     (SP)+           ;;POP  <CR><LF> EQUIV
010666  104401                          TYPE                    ;;TYPE A CR AND LF
010670  001171                          $CRLF
010672  105037  011026                  CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
010676  000755                          BR      2$              ;;GET NEXT CHARACTER
010700  004737  010762          5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
010704  123726  001156          6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
010710  001350                          BNE     2$              ;;IF NO GO GET NEXT CHAR.
010712  013746  001154                  MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
                                                                ;;AND THE NULL CHAR.
010716  105366  000001          7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
010722  002770                          BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
```

```
 2143  010724  004737  010762                  JSR     PC,$TYPEC       ;;GO TYPE A NULL
 2144  010730  105337  011026                  DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
 2145  010734  000770                          BR      7$              ;;LOOP
 2146
 2147                                  ;HORIZONTAL TAB PROCESSOR
 2148
 2149  010736  112716  000040          8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
 2150  010742  004737  010762          9$:     JSR     PC,$TYPEC       ;;TYPE A SPACE
 2151  010746  132737  000007  011026          BITB    #7,$CHAR.NT     ;;BRANCH IF NOT AT
 2152  010754  001372                          BNE     9$              ;;TAB STOP
 2153  010756  005726                          TST     (SP)+           ;;POP SPACE OFF STACK
 2154  010760  000724                          BR      2$              ;;GET NEXT CHARACTER
 2155  010762  105777  170162          $TYPEC: TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
 2156  010766  100375                          BPL     $TYPEC
 2157  010770  116677  000002  170154          MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
 2158  010776  122766  000015  000002          CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
 2159  011004  001003                          BNE     1$              ;;BRANCH IF NO
 2160  011006  105037  011026                  CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
 2161  011012  000406                          BR      $TYPEX          ;;EXIT
 2162  011014  122766  000012  000002  1$:     CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
 2163  011022  001402                          BEQ     $TYPEX          ;;BRANCH IF YES
 2164  011024  105227                          INCB    (PC)+           ;;COUNT THE CHARACTER
 2165  011026  000000          $CHARCNT:.WORD  0               ;;CHARACTER COUNT STORAGE
 2166  011030  000207          $TYPEX: RTS     PC
 2167
```

```
                                        .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
                                        ;;**************************************************************
                                        ;;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
                                        ;;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
                                        ;;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
                                        ;;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
                                        ;;*REPLACED WITH SPACES.
                                        ;;*CALL:
                                        ;;*      MOV     NUM,-(SP)       ;;PUT THE BINARY NUMBER ON THE STACK
                                        ;;*      TYPOS                   ;;GO TO THE ROUTINE
                                        ;;
011032                                  $TYPOS:
011032  010046                                 MOV     R0,-(SP)        ;;PUSH R0 ON STACK
011034  010146                                 MOV     R1,-(SP)        ;;PUSH R1 ON STACK
011036  010246                                 MOV     R2,-(SP)        ;;PUSH R2 ON STACK
011040  010346                                 MOV     R3,-(SP)        ;;PUSH R3 ON STACK
011042  010546                                 MOV     R5,-(SP)        ;;PUSH R5 ON STACK
011044  012746  020200                        MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
011050  016605  000020                        MOV     20(SP),R5       ;;GET THE INPUT NUMBER
011054  100004                                BPL     1$              ;;BR IF INPUT IS POS.
011056  005405                                NEG     R5              ;;MAKE THE BINARY NUMBER POS.
011060  112766  000055  000001                MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
011066  005000                        1$:     CLR     R0              ;;ZERO THE CONSTANTS INDEX
011070  012703  011246                        MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
011074  112723  000040                        MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
011100  005002                        2$:     CLR     R2              ;;CLEAR THE BCD NUMBER
011102  016001  011236                        MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
011106  160105                        3$:     SUB     R1,R5           ;;FORM THIS BCD DIGIT
011110  002402                                BLT     4$              ;;BR IF DONE
011112  005202                                INC     R2              ;;INCREASE THE BCD DIGIT BY 1
011114  000774                                BR      3$
011116  060105                        4$:     ADD     R1,R5           ;;ADD BACK THE CONSTANT
011120  005702                                TST     R2              ;;CHECK IF BCD DIGIT=0
011122  001002                                BNE     5$              ;;FALL THROUGH IF 0
011124  105716                                TSTB    (SP)            ;;STILL DOING LEADING 0'S?
011126  100407                                BMI     7$              ;;BR IF YES
011130  106316                        5$:     ASLB    (SP)            ;;MSD?
011132  103003                                BCC     6$              ;;BR IF NO
011134  116663  000001  177777                MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
011142  052702  000060                6$:     BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
011146  052702  000040                7$:     BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
011152  110223                                MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
011154  005720                                TST     (R0)+           ;;JUST INCREMENTING
011156  020027  000010                        CMP     R0,#10          ;;CHECK THE TABLE INDEX
011162  002746                                BLT     2$              ;;GO DO THE NEXT DIGIT
011164  003002                                BGT     8$              ;;GO TO EXIT
011166  010502                                MOV     R5,R2           ;;GET THE LSD
011170  000764                                BR      6$              ;;GO CHANGE TO ASCII
011172  105726                        8$:     TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
011174  100003                                BPL     9$              ;;BR IF NO
011176  116663  177777  177776                MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
011204  105013                        9$:     CLRB    (R3)            ;;SET THE TERMINATOR
011206  012605                                MOV     (SP)+,R5        ;;POP STACK INTO R5
```

```
2222  011210  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
2223  011212  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
2224  011214  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
2225  011216  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
2226  011220  104401  011246                  TYPE    ,$DBLK          ;;NOW TYPE THE NUMBER
2227  011224  016666  000002  000004          MOV     2(SP),4(SP)     ;;ADJUST THE STACK
2228  011232  012616                          MOV     (SP)+,(SP)
2229  011234  000002                          RTI                     ;;RETURN TO USER
2230  011236  023420                  $DTBL:  10000.
2231  011240  001750                          1000.
2232  011242  000144                          100.
2233  011244  000012                          10.
2234  011246  000004                  $DBLK:  .BLKW   4
```

```
                                        .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE

                                        ;;**************************************************************
                                        ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
                                        ;*OCTAL (ASCII) NUMBER AND TYPE IT.
                                        ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
                                        ;*CALL:
                                        ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                                        ;*      TYPOS                   ;;CALL FOR TYPEOUT
                                        ;*      .BYTE   N               ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
                                        ;*      .BYTE   M               ;;M=1 OR 0
                                        ;*                                      ;;1=TYPE LEADING ZEROS
                                        ;*                                      ;;0=SUPPRESS LEADING ZEROS
                                        ;*
                                        ;*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
                                        ;*$TYPOS OR $TYPOC
                                        ;*CALL:
                                        ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                                        ;*      TYPON                   ;;CALL FOR TYPEOUT
                                        ;*
                                        ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
                                        ;*CALL:
                                        ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                                        ;*      TYPOC                   ;;CALL FOR TYPEOUT

011256  017646  000000          $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
011262  116637  000001  011501          MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
011270  112637  011503                  MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
011274  062716  000002                  ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
011300  000406                          BR      $TYPON
011302  112737  000001  011501  $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
011310  112737  000006  011503          MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
011316  112737  000005  011500  $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
011324  010346                          MOV     R3,-(SP)        ;;SAVE R3
011326  010446                          MOV     R4,-(SP)        ;;SAVE R4
011330  010546                          MOV     R5,-(SP)        ;;SAVE R5
011332  113704  011503                  MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
011336  005404                          NEG     R4
011340  062704  000006                  ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
011344  110437  011502                  MOVB    R4,$OMODE       ;;SAVE IT FOR USE
011350  113704  011501                  MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
011354  016605  000012                  MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
011360  005003                          CLR     R3              ;;CLEAR THE OUTPUT WORD
011362  006105                  1$:     ROL     R5              ;;ROTATE MSB INTO "C"
011364  000404                          BR      3$              ;;GO DO MSB
011366  006105                  2$:     ROL     R5              ;;FORM THIS DIGIT
011370  006105                          ROL     R5
011372  006105                          ROL     R5
011374  010503                          MOV     R5,R3
011376  006103                  3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
011400  105337  011502                  DECB    $OMODE          ;;TYPE THIS DIGIT?
011404  100016                          BPL     7$              ;;BR IF NO
011406  042703  177770                  BIC     #177770,R3      ;;GET RID OF JUNK
011412  001002                          BNE     4$              ;;TEST FOR 0
```

```
011414  005704                          TST     R4              ;;SUPPRESS THIS 0?
011416  001403                          BEQ     5$              ;;BR IF YES
011420  005204                  4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
011422  052703  000060                  BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
011426  052703  000040          5$:     BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
011432  110337  011476                  MOVB    R3,@#8$         ;;SAVE FOR TYPING
011436  104401  011476                  TYPE    ,@#8$           ;;GO TYPE THIS DIGIT
011442  105337  011500          7$:     DECB    @#$OCNT         ;;COUNT BY 1
011446  003347                          BGT     2$              ;;BR IF MORE TO DO
011450  002402                          BLT     6$              ;;BR IF DONE
011452  005204                          INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
011454  000744                          BR      2$              ;;GO DO THE LAST DIGIT
011456  012605                  6$:     MOV     (SP)+,R5        ;;RESTORE R5
011460  012604                          MOV     (SP)+,R4        ;;RESTORE R4
011462  012603                          MOV     (SP)+,R3        ;;RESTORE R3
011464  016666  000002  000004          MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
011472  012616                          MOV     (SP)+,(SP)
011474  000002                          RTI                     ;;RETURN
011476  000                     8$:     .BYTE   0               ;;STORAGE FOR ASCII DIGIT
011477  000                             .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
011500  000                     $OCNT:  .BYTE   0               ;;OCTAL DIGIT COUNTER
011501  000                     $OFILL: .BYTE   0               ;;ZERO FILL SWITCH
011502  000000                  $OMODE: .WORD   0               ;;NUMBER OF DIGITS TO TYPE
```

```
                                        .SBTTL  BINARY TO ASCII AND TYPE ROUTINE
                                        ;;*****************************************************************
                                        ;;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 16-BIT
                                        ;;*BINARY-ASCII NUMBER AND TYPE IT.
                                        ;;*CALL:
                                        ;;*     MOV     NUMBER,-(SP)    ;;NUMBER TO BE TYPED
                                        ;;*     TYPBN                   ;;TYPE IT
011504  010146                  $TYPBN: MOV     R1,-(SP)        ;;SAVE R1 ON THE STACK
011506  016601  000006                  MOV     6(SP),R1        ;;GET THE INPUT NUMBER
011512  000261                          SEC                     ;;SET "C" SO CAN KEEP TRACK OF THE NUMBER OF BITS
011514  112737  000060  011556  1$:     MOVB    #'0,$BIN        ;;SET CHARACTER TO AN ASCII "0".
011522  006101                          ROL     R1              ;;GET THIS BIT
011524  001406                          BEQ     2$              ;;DONE?
011526  105537  011556                  ADCB    $BIN            ;;NO--SET THE CHARACTER EQUAL TO THIS BIT
011532  104401  011556                  TYPE    ,$BIN           ;;GO TYPE THIS BIT
011536  000241                          CLC                     ;;CLEAR "C" SO CAN KEEP TRACK OF BITS
011540  000765                          BR      1$              ;;GO DO THE NEXT BIT
011542  012601                  2$:     MOV     (SP)+,R1        ;;POP THE STACK INTO R1
011544  016666  000002  000004          MOV     2(SP),4(SP)     ;;ADJUST THE STACK
011552  012616                          MOV     (SP)+,(SP)
011554  000002                          RTI                     ;;RETURN TO USER
011556  000     000             $BIN:   .BYTE   0,0             ;;STORAGE FOR ASCII CHAR. AND TERMINATOR
```

```
                                        .SBTTL  TTY INPUT ROUTINE
                                        ;;*****************************************************************
                                        ;
                                        .ENABL  LSB
                                        ;;*****************************************************************
                                        ;;#SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
                                        ;#ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
                                        ;#SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
                                        ;#WHEN OPERATING IN TTY FLAG MODE.
011560  022737  000176  001140  $CKSWR: CMP     @#SWREG,SWR     ;;IS THE SOFT-SWR SELECTED?
011566  001074                        BNE     1$$             ;;BRANCH IF NO
011570  105777  167350                TSTB    @#TKS           ;;CHAR THERE?
011574  100071                        BPL     1$$             ;;IF NO, DON'T WAIT AROUND
011576  117746  167344                MOVB    @#TKB,-(SP)     ;;SAVE THE CHAR
011602  042716  177600                BIC     #^C177,(SP)     ;;STRIP-OFF THE ASCII
011606  022726  000007                CMP     #7,(SP)+        ;;IS IT A CONTROL G?
011612  001062                        BNE     1$$             ;;NO, RETURN TO USER
011614  123727  001134  000001        CMPB    $AUTOB,#1       ;;ARE WE RUNNING IN AUTO-MODE?
011622  001456                        BEQ     1$$             ;;BRANCH IF YES
                                        ;
011624  104401  012305                TYPE    ,$CNTLG         ;;ECHO THE CONTROL-G (^G)
011630  104401  012312        $GTSWR: TYPE    ,$MSWR          ;;TYPE CURRENT CONTENTS
011634  013746  000176                MOV     $WREG,-(SP)     ;;SAVE $WREG FOR TYPEOUT
011640  104402                        TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
011642  104401  012323                TYPE    ,$MNEW          ;;PROMPT FOR NEW SWR
011646  005046                1$$:    CLR     -(SP)           ;;CLEAR COUNTER
011650  005046                        CLR     -(SP)           ;;THE NEW SWR
011652  105777  167266        7$:     TSTB    @#TKS           ;;CHAR THERE?
011656  100375                        BPL     7$              ;;IF NOT TRY AGAIN
                                        ;
011660  117746  167262                MOVB    @#TKB,-(SP)     ;;PICK UP CHAR
011664  042716  177600                BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
                                        ;
                                        ;
011670  021627  000025        9$:     CMP     (SP),#25        ;;IS IT A CONTROL-U?
011674  001005                        BNE     10$             ;;BRANCH IF NOT
011676  104401  012300                TYPE    ,$CNTLU         ;;YES, ECHO CONTROL-U (^U)
011702  062706  000006        20$:    ADD     #6,SP           ;;IGNORE PREVIOUS INPUT
011706  000757                        BR      1$$             ;;LET'S TRY IT AGAIN
                                        ;
                                        ;
011710  021627  000015        10$:    CMP     (SP),#15        ;;IS IT A <CR>?
011714  001022                        BNE     16$             ;;BRANCH IF NO
011716  005766  000004                TST     4(SP)           ;;YES, IS IT THE FIRST CHAR?
011722  001403                        BEQ     11$             ;;BRANCH IF YES
011724  016677  000002  167206        MOV     2(SP),@#SWR     ;;SAVE NEW SWR
011732  062706  000006        11$:    ADD     #6,SP           ;;CLEAR UP STACK
011736  104401  001171        14$:    TYPE    ,$CRLF          ;;ECHO <CR> AND <LF>
011742  123727  001135  000001        CMPB    $INTAG,#1       ;;RE-ENABLE TTY KBD INTERRUPTS?
011750  001003                        BNE     15$             ;;BRANCH IF NOT
011752  012777  000100  167164        MOV     #100,@#TKS      ;;RE-ENABLE TTY KBD INTERRUPTS
011760  000002                15$:    RTI                     ;;RETURN
```

```
2390  011762  004737  010762          16$:    JSR     PC,$TYPEC       ;;ECHO CHAR
2391  011766  021627  000060                  CMP     (SP),#60        ;;CHAR < 0?
2392  011772  002450                          BLT     18$             ;;BRANCH IF YES
2393  011774  021627  000067                  CMP     (SP),#67        ;;CHAR > 7?
2394  012000  003015                          BGT     18$             ;;BRANCH IF YES
2395  012002  042726  000060                  BIC     #60,(SP)+       ;;STRIP-OFF ASCII
2396  012006  005766  000002                  TST     2(SP)           ;;IS THIS THE FIRST CHAR
2397  012012  001403                          BEQ     17$             ;;BRANCH IF YES
2398  012014  006316                          ASL     (SP)            ;;NO, SHIFT PRESENT
2399  012016  006316                          ASL     (SP)            ;;   CHAR OVER TO MAKE
2400  012020  006316                          ASL     (SP)            ;;   ROOM FOR NEW ONE.
2401  012022  005266  000002          17$:    INC     2(SP)           ;;KEEP COUNT OF CHAR
2402  012026  050616  177776                  BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
2403  012032  000707                          BR      7$              ;;GET THE NEXT ONE
2404  012034  104401  001170          18$:    TYPE    ,$QUES          ;;TYPE ?<CR><LF>
2405  012040  000720                          BR      20$             ;;SIMULATE CONTROL-U
2406                                  .DSABL  LSB
2407
2408
2409                                  ;;**************************************************************
2410                                  ;;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
2411                                  ;;*CALL:
2412                                  ;;*     RDCHR                   ;;INPUT A SINGLE CHARACTER FROM THE TTY
2413                                  ;;*     RETURN HERE             ;;CHARACTER IS ON THE STACK
2414                                  ;;*                             ;;WITH PARITY BIT STRIPPED OFF
2415                                  ;;
2416
2417  012042  011646                  $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC
2418  012044  016666  000004  000002          MOV     4(SP),2(SP)     ;;SAVE THE PS
2419  012052  105777  167066          1$:     TSTB    @#TKS           ;;WAIT FOR
2420  012056  100375                          BPL     1$              ;;A CHARACTER
2421  012060  117766  167062  000004          MOVB    @#TKB,4(SP)     ;;READ THE TTY
2422  012066  042766  177600  000004          BIC     #^C(177),4(SP)  ;;GET RID OF JUNK IF ANY
2423  012074  026627  000004  000023          CMP     4(SP),#23       ;;IS IT A CONTROL-S?
2424  012102  001013                          BNE     3$              ;;BRANCH IF NO
2425  012104  105777  167034          2$:     TSTB    @#TKS           ;;WAIT FOR A CHARACTER
2426  012110  100375                          BPL     2$              ;;LOOP UNTIL ITS THERE
2427  012112  117746  167030                  MOVB    @#TKB,-(SP)     ;;GET CHARACTER
2428  012116  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
2429  012122  022627  000021                  CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
2430  012126  001366                          BNE     2$              ;;IF NOT DISCARD IT
2431  012130  000750                          BR      1$              ;;YES, RESUME
2432  012132  026627  000004  000140  3$:     CMP     4(SP),#140      ;;IS IT UPPER CASE?
2433  012140  002407                          BLT     4$              ;;BRANCH IF YES
2434  012142  026627  000004  000175          CMP     4(SP),#175      ;;IS IT A SPECIAL CHAR?
2435  012150  003003                          BGT     4$              ;;BRANCH IF YES
2436  012152  042766  000040  000004          BIC     #40,4(SP)       ;;MAKE IT UPPER CASE
2437  012160  000002                  4$:     RTI                     ;;GO BACK TO USER
2438                                  ;;**************************************************************
2439                                  ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
2440                                  ;*CALL:
2441                                  ;*     RDLIN                   ;;INPUT A STRING FROM THE TTY
2442                                  ;*     RETURN HERE             ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
2443                                  ;*                             ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
```

```
012162  010346                    $RDLIN: MOV     R3,-(SP)        ;;SAVE R3
012164  012703  012270            1$:     MOV     #$TTYIN,R3      ;;GET ADDRESS
012170  022703  012300            2$:     CMP     #$TTYIN+8.,R3   ;;BUFFER FULL?
012174  101405                            BLOS    4$              ;;BR IF YES
012176  104411                            RDCHR                   ;;GO READ ONE CHARACTER FROM THE TTY
012200  112613                            MOVB    (SP)+,(R3)      ;;GET CHARACTER
012202  122713  000177            10$:    CMPB    #177,(R3)       ;;IS IT A RUBOUT
012206  001003                            BNE     3$              ;;SKIP IF NOT
012210  104401  001170            4$:     TYPE    ,$QUES          ;;TYPE A '?'
012214  000763                            BR      1$              ;;CLEAR THE BUFFER AND LOOP
012216  111337  012266            3$:     MOVB    (R3),9$         ;;ECHO THE CHARACTER
012222  104401  012266                    TYPE    ,9$
012226  122723  000015                    CMPB    #15,(R3)+       ;;CHECK FOR RETURN
012232  001356                            BNE     2$              ;;LOOP IF NOT RETURN
012234  10[illegible]63  177777                    CLRB    -1(R3)          ;;CLEAR RETURN (THE 15)
012240  104401  001172                    TYPE    ,$LF            ;;TYPE A LINE FEED
012244  012603                            MOV     (SP)+,R3        ;;RESTORE R3
012246  011646                            MOV     (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
01[illegible]0  01[illegible]6  000004  000002          MOV     4(SP),2(SP)     ;;      FIRST ASCII CHARACTER ON IT
01[illegible]  01[illegible]  012270  000004          MOV     #$TTYIN,4(SP)
012[illegible]64  000002                            RTI                     ;;RETURN
012[illegible]6     000                   9$:     .BYTE   0               ;;STORAGE FOR ASCII CHAR. TO TYPE
01[illegible]7     000                           .BYTE   0               ;;TERMINATOR
012270  000010                    $TTYIN: .BLKB   8.              ;;RESERVE 8 BYTES FOR TTY INPUT
012300  0[illegible]536  0[illegible]15     000  $CNTLU: .ASCIZ  /↑U/<15><12>    ;;CONTROL "U"
012305     136  [illegible]7  000012  $[illegible]NTLG: .ASCIZ  /↑G/<15><12>    ;;CONTROL "G"
012312  005015  05[illegible]3  020122  $MSWR:  .ASCIZ  <15><12>/SWR = /
012320  0[illegible]75     0[illegible]0
012323     040  047040  053505  $MNEW:  .ASCIZ  /  NEW = /
012330  036440  000040
```

```
                                        .SBTTL  READ AN OCTAL NUMBER FROM THE TTY
                                        ;;*****************************************************************
                                        ;;*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
                                        ;;*CHANGE IT TO BINARY.
                                        ;;*THE INPUT CHARACTERS WILL BE CHECKED TO INSURED THEY ARE LEGAL
                                        ;;*OCTAL DIGITS. IF AN ILLEGAL CHARACTER IS READ A "?" WILL BE TYPED
                                        ;;*FOLLOWED BY A CARRIAGE RETURN-LINE FEED. THE COMPLETE NUMBER MUST
                                        ;;*THEN BE RETYPED. THE INPUT IS TERMINATED BY TYPING A CARRIAGE RETURN.
                                        ;;*CALL:
                                        ;;*      RDOCT                   ;;READ AN OCTAL NUMBER
                                        ;;*      RETURN HERE             ;;LOW ORDER BITS ARE ON TOP OF THE STACK
                                        ;;*                              ;;HIGH ORDER BITS ARE IN $HIOCT
012394  011646                  $RDOCT: MOV     (SP),-(SP)              ;;PROVIDE SPACE FOR THE
012396  016666  000004  000002          MOV     4(SP),2(SP)             ;;INPUT NUMBER
012404  010046                          MOV     R0,-(SP)                ;;PUSH R0 ON STACK
012406  010146                          MOV     R1,-(SP)                ;;PUSH R1 ON STACK
012410  010246                          MOV     R2,-(SP)                ;;PUSH R2 ON STACK
012412  104412                  1$:     RDLIN                           ;;READ AN ASCIZ LINE
012414  012600                          MOV     (SP)+,R0                ;;GET ADDRESS OF 1ST CHARACTER
012416  010037  012462                  MOV     R0,5$                   ;;AND SAVE IT
012422  005001                          CLR     R1                      ;;CLEAR DATA WORD
012424  005002                          CLR     R2
012426  112046                  2$:     MOVB    (R0)+,-(SP)             ;;PICKUP THIS CHARACTER
012430  001420                          BEQ     3$                      ;;IF ZERO GET OUT
012432  122716  000060                  CMPB    #'0,(SP)                ;;MAKE SURE THIS CHARACTER
012436  003026                          BGT     4$                      ;;IS AN OCTAL DIGIT
012440  122716  000067                  CMPB    #'7,(SP)
012444  002423                          BLT     4$
012446  006301                          ASL     R1                      ;;*2
012450  006102                          ROL     R2
012452  006301                          ASL     R1                      ;;*4
012454  006102                          ROL     R2
012456  006301                          ASL     R1                      ;;*8
012460  006102                          ROL     R2
012462  042716  177770                  BIC     #^C7,(SP)               ;;STRIP THE ASCII JUNK
012466  060116                          ADD     (SP)+,R1                ;;ADD IN THIS DIGIT
012470  000756                          BR      2$                      ;;LOOP
012472  005726                  3$:     TST     (SP)+                   ;;CLEAN TERMINATOR FROM STACK
012474  010066  000012                  MOV     R1,12(SP)               ;;SAVE THE RESULT
012500  013746  012472                  MOV     R2,$HIOCT
012504  012602                          MOV     (SP)+,R2                ;;POP STACK INTO R2
012506  012601                          MOV     (SP)+,R1                ;;POP STACK INTO R1
012510  012600                          MOV     (SP)+,R0                ;;POP STACK INTO R0
012512  000002                          RTI                             ;;RETURN
012514  005726                  4$:     TST     (SP)+                   ;;CLEAN PARTIAL FROM STACK
012516  105010                          CLRB    (R0)                    ;;SET A TERMINATOR
012520  104401                          TYPE                            ;;TYPE UP THRU THE BAD CHAR.
012522  000000                  5$:     .WORD   0
012524  104401  001170                  TYPE    ,SQUES                  ;;"?" "CR" & "LF"
012530  000730                          BR      1$                      ;;TRY AGAIN
012532  000000                  $HIOCT: .WORD   0                       ;;HIGH ORDER BITS GO HERE
```

```
                                        .SBTTL  POWER DOWN AND UP ROUTINES
                                        ;;*****************************************************************
                                        ;POWER DOWN ROUTINE
012474  012737  012634  000024  $PWRDN: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
012502  012737  000340  000026          MOV     #340,@#PWRVEC+2  ;;PRIO:7
012510  010046                          MOV     R0,-(SP)         ;;PUSH R0 ON STACK
012512  010146                          MOV     R1,-(SP)         ;;PUSH R1 ON STACK
012514  010246                          MOV     R2,-(SP)         ;;PUSH R2 ON STACK
012516  010346                          MOV     R3,-(SP)         ;;PUSH R3 ON STACK
        010446                          MOV     R4,-(SP)         ;;PUSH R4 ON STACK
        010546                          MOV     R5,-(SP)         ;;PUSH R5 ON STACK
012524  017746  166410                  MOV     @#SR,-(SP)       ;;PUSH @#SR ON STACK
        010637  012540                  MOV     SP,$SAVR6        ;;SAVE SP
        012737  012546  000024          MOV     #$PWRUP,@#PWRVEC ;;SET UP VECTOR
012542  000000                          HALT
012544  000776                          BR      .-2              ;;HANG UP
                                        ;;*****************************************************************
                                        ;POWER UP ROUTINE
012546  012737  012634  000024  $PWRUP: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
        013706  012540                  MOV     $SAVR6,SP        ;;GET SP
        005037  012640                  CLR     $SAVR6           ;;WAIT LOOP FOR THE TTY
        005237  012640          1$:     INC     $SAVR6           ;;WAIT FOR THE INC
001370  001375                          BNE     1$               ;;OF  WORD
012572  012677  166342                  MOV     (SP)+,@#SR       ;;POP STACK INTO @#SR
        012605                          MOV     (SP)+,R5         ;;POP STACK INTO R5
        012504                          MOV     (SP)+,R4         ;;POP STACK INTO R4
        012503                          MOV     (SP)+,R3         ;;POP STACK INTO R3
        012502                          MOV     (SP)+,R2         ;;POP STACK INTO R2
        012501                          MOV     (SP)+,R1         ;;POP STACK INTO R1
012610  012600                          MOV     (SP)+,R0         ;;POP STACK INTO R0
012512  012737  012474  000024          MOV     #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
012520  012737  000340  000C26          MOV     #340,@#PWRVEC+2  ;;PRIO:7
012626  104401                          TYPE                     ;REPORT THE POWER FAILURE
012630  012642          $PWRMG: .WORD   $POWER                   ;;POWER FAIL MESSAGE POINTER
        000002                  RTI
012634  000000          $ILLUP: HALT                             ;;THE POWER UP SEQUENCE WAS STARTED
012636  000776                  BR      .-2                      ;;  BEFORE THE POWER DOWN WAS COMPLETE
012640  000000          $SAVR6: 0                                ;;PUT THE SP HERE
012642  005015  047520  042527  $POWER: .ASCIZ  <15><12>"POWER"
012650  000122
                                .EVEN
```

```
                                        .SBTTL  APT COMMUNICATIONS ROUTINE
                                        ;;*****************************************************************
012652  112737  000001  013116  $ATY1:  MOVB    #1,$FFLG        ;;TO REPORT FATAL ERROR
012660  112737  000001  013114  $ATY3:  MOVB    #1,$MFLG        ;;TO TYPE A MESSAGE
012666  000403                        BR      $ATYC
012670  112737  000001  013116  $ATY4:  MOVB    #1,$FFLG        ;;TO ONLY REPORT FATAL ERROR
012676                          $ATYC:
012676  010046                        MOV     R0,-(SP)        ;;PUSH R0 ON STACK
012700  010146                        MOV     R1,-(SP)        ;;PUSH R1 ON STACK
012702  105737  013114                TSTB    $MFLG           ;;SHOULD TYPE A MESSAGE?
012706  001450                        BEQ     5$              ;;IF NOT:  BR
012710  122737  000001  001214        CMPB    #APTENV,$ENV    ;;OPERATING UNDER APT?
012716  001031                        BNE     3$              ;;IF NOT:  BR
012720  132737  000100  001215        BITB    #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
012726  001425                        BEQ     3$              ;;IF NOT:  BR
012730  017600  000004                MOV     @4(SP),R0       ;;GET MESSAGE ADDR.
012734  062766  000002  000004        ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
012742  005737  001174          1$:   TST     $MSGTYPE        ;;SEE IF DONE W/ LAST XMISSION?
012746  001375                        BNE     1$              ;;IF NOT:  WAIT
012750  010037  001210                MOV     R0,$MSGAD       ;;PUT ADDR IN MAILBOX
012754  105720                  2$:   TSTB    (R0)+           ;;FIND END OF MESSAGE
012756  001376                        BNE     2$
012760  163700  001210                SUB     $MSGAD,R0       ;;SUB START OF MESSAGE
012764  006200                        ASR     R0              ;;GET MESSAGE LNGTH IN WORDS
012766  010037  001212                MOV     R0,$MSGLGT      ;;PUT LENGTH IN MAILBOX
012772  012737  000004  001174        MOV     #4,$MSGTYPE     ;;TELL APT TO TAKE MSG.
013000  000413                        BR      5$
013002  017637  000004  013026  3$:   MOV     @4(SP),4$       ;;PUT MSG ADDR IN JSR LINKAGE
013010  062766  000002  000004        ADD     #2,4(SP)                ;;BUMP RETURN ADDRESS
013016  013746  177776                MOV     177776,-(SP)    ;;PUSH 177776 ON STACK
013022  004737  010550                JSR     PC,$TYPE        ;;CALL TYPE MACRO
013036  000000                  4$:   .WORD   0
013030                          5$:
013030  105737  013116          10$:  TSTB    $FFLG           ;;SHOULD REPORT FATAL ERROR?
013034  001416                        BEQ     12$             ;;IF NOT:  BR
013036  005737  001214                TST     $ENV            ;;RUNNING UNDER APT?
013042  001413                        BEQ     12$             ;;IF NOT:  BR
013044  005737  001174          11$:  TST     $MSGTYPE        ;;FINISHED LAST MESSAGE?
013050  001375                        BNE     11$             ;;IF NOT:  WAIT
013052  017637  000004  001176        MOV     @4(SP),$FATAL   ;;GET ERROR #
013060  062766  000002  000004        ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
013066  005237  001174                INC     $MSGTYPE        ;;TELL APT TO TAKE ERROR
013072  105037  013116          12$:  CLRB    $FFLG           ;;CLEAR FATAL FLAG
013076  105037  013115                CLRB    $LFLG           ;;CLEAR LOG FLAG
013102  105037  013114                CLRB    $MFLG           ;;CLEAR MESSAGE FLAG
013106  012601                        MOV     (SP)+,R1        ;;POP STACK INTO R1
013110  012600                        MOV     (SP)+,R0        ;;POP STACK INTO R0
013112  000207                        RTS     PC              ;;RETURN
013114     000          $MFLG:  .BYTE   0               ;;MESSG. FLAG
013115     000          $LFLG:  .BYTE   0               ;;LOG FLAG
013116     000          $FFLG:  .BYTE   0               ;;FATAL FLAG
013120                          .EVEN
000200                  APTSIZE=200
```

```
  2625            000001                  APTENV=001
  2626            000100                  APTSPOOL=100
  2627            000040                  APTCSUP=040
```

```
                                        .SBTTL  TRAP DECODER

                                        ;;**********************************************************************
                                        ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
                                        ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
                                        ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
                                        ;*GO TO THAT ROUTINE.

013120  010046                  STRAP:  MOV     R0,-(SP)        ;;SAVE R0
013122  016500  000002                  MOV     2(SP),R0        ;;GET TRAP ADDRESS
013126  005340                          TST     -(R0)           ;;BACKUP BY 2
013130  111000                          MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
013132  006300                          ASL     R0              ;;POSITION FOR INDEXING
013134  016000  013154                  MOV     STRPAD(R0),R0   ;;INDEX TO TABLE
013140  000200                          RTS     R0              ;;GO TO ROUTINE


                                        ;;THIS IS USE TO HANDLE THE "GETPRI" MACRO

013142  011646                  STRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
013144  016666  000004  000002          MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
013152  000002                          RTI                     ;;RESTORE THE PSW

                                        .SBTTL  TRAP TABLE

                                        ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
                                        ;*BY THE "TRAP" INSTRUCTION.

                                        ;       ROUTINE
                                        ;       -------

013154  013142                  STRPAD: .WORD   STRAP2
013156  010550                          $TYPE   ;;CALL=TYPE     TRAP+1(104401)  TTY TYPEOUT ROUTINE
013160  011302                          $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
013162  011256                          $TYPOS  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
013164  011316                          $TYPON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
013166  011032                          $TYPDS  ;;CALL=TYPDS    TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
013170  011504                          $TYPBN  ;;CALL=TYPBN    TRAP+6(104406)  TYPE BINARY (ASCII) NUMBER

013172  011630                          $GTSWR  ;;CALL=GTSWR    TRAP+7(104407)  GET SOFT-SWR SETTING

013174  011560                          $CKSWR  ;;CALL=CKSWR    TRAP+10(104410) TEST FOR CHANGE IN SOFT-SWR
013176  012042                          $RDCHR  ;;CALL=RDCHR    TRAP+11(104411) TTY TYPEIN CHARACTER ROUTINE
013200  012162                          $RDLIN  ;;CALL=RDLIN    TRAP+12(104412) TTY TYPEIN STRING ROUTINE
013202  012334                          $RDOCT  ;;CALL=RDOCT    TRAP+13(104413) READ AN OCTAL NUMBER FROM TTY
```

```
                                        .SBTTL  ERROR MESSAGES AND MISCELLANEOUS.
                                        ;;*****************************************************************
013204 047111 040526 044514 A1:         .ASCII  /INVALID RAM ADDRESS RANGE./<CRLF>
013212 020104 040522 020115
013220 042101 051104 051505
013226 020123 040522 043516
013234 027105      200
013237      116 054105 020115           .ASCIZ  /NEXM OR NOT ON 256 WORD BOUNDARY./
013244 051117 047040 052117
013252 047440 020116 032462
013260 042040 047527 042122
013266 042040 047517 042116
013274 051101 052131      000
013301      111 053116 046101 M2:       .ASCII  /INVALID PROM ADDRESS RANGE./<CRLF>
013306 042111 052040 047522
013314 052115 042101 051104
013322 052105 040123 040522
013330 042040 047105      200
013335      116 054105 026115           .ASCIZ  /NEXM, INCORRECT SIZE, OR NOT ON 1K BOUNDARY./
013342 044440 041516 051117
013350 042040 052103 051440
013356 042111 047105 047440
013364 042122 047516 020124
013372 047117 030440 052113
013400 047502 047125 040504
013406 054522 000056
013412 047111 052056 044514 M3:         .ASCIZ  /INVALID DUMP RANGE./
013420 042104 052056 050115
013426 051040 047101 042507
013434 052056
013436 042040 052115 042522 EM1:        .ASCIZ  /RAM READ-WRITE ERROR./
013444 042101 053455 044522
013452 042524 042440 051122
013460 052117 052040
013464 020124 020124 004443 DH1:        .ASCIZ  /TEST#  ERRPC   ADDR    GOOD    BAD/
013472 051105 020122 004503
013500 042101 051104 043411
013506 047517 004504 040502
013514 000104
013516                                  .EVEN
013516 001200 001116 001122 DT1:        .WORD   $TESTN,$ERRPC,$BDADR,$GDDAT,$BDDAT,0
013524 001124 001126 000000
013532 040522 020115 042101 EM2:        .ASCIZ  /RAM ADDRESS ERROR./
013540 051104 051505 020123
013546 051105 047522 027122
013554      000
013557      124 051505 021524 DH2:      .ASCIZ  /TEST#  ERRPC   ADDR    GOOD    BAD/
013562 042411 051122 041520
013570 040411 042104 004522
013576 047507 042117 041011
013604 042101      000
013610                                  .EVEN
013610 001200 001116 001122 DT2:        .WORD   $TESTN,$ERRPC,$BDADR,$GDDAT,$BDDAT,0
013616 001124 001126 000000
```

```
 013624  040522  020115  040506  EM3:    .ASCIZ  /RAM FAST READ ERROR./
 013632  052123  051040  04[illegible]5
 013640  020104  051105  047522
 013646  027122     000
 013651     124  051505  021524  DH3:    .ASCIZ  /TEST#  ERRPC   ADDR    GOOD    RAMDAT  BUFDAT/
 013656  042411  051122  041520
 013664  040411  042104  004522
 013672  047507  042117  051011
 013700  046501  040504  004524
 013706  052502  042106  052101
 013714     000
         013716                          .EVEN
 013716  001200  001116  001120  DT3:    .WORD   $TESTN,$ERRPC,$GDADR,$GDADR,$BDADR,$BDDAT,0
 013724  001120  001122  001126
 013732  000000
 013734  040522  020115  046123  EM4:    .ASCIZ  /RAM SLIDING BIT.  ADDR-2 WAS ALTERED./
 013742  042111  047111  020107
 013750  044502  027124  0[illegible]040
 013756  042101  051104  0310[illegible]5
 013764  053440  051501  04[illegible]440
 013772  052114  051105  042105
 014000  000056
 014002  042524  052123  004443  DH4:    .ASCIZ  /TEST#  ERRPC   ADDR    PATRN   ADR-2   GOOD    BAD/
 014010  051105  0[illegible]122  004503
 014016  042101  051104  0[illegible]011
 014024  052101  047122  040411
 014032  051104  031055  043411
 014040  047517  004504  040502
 014046  000104
                                         .EVEN
 014050  001200  001116  001120  DT4:    .WORD   $TESTN,$ERRPC,$GDADR,$GDDAT,$BDADR,FILLR,$BDDAT,0
 014056  001124  001122  005060
 014064  001126  000000
 014070  040522  020115  046123  EM5:    .ASCIZ  /RAM SLIDING BIT.  ADDR+2 WAS ALTERED./
 014076  042111  047111  020107
 014104  044502  027124  020040
 014112  042101  051104  031053
 014120  053440  051501  040440
 014126  052114  051105  042105
 014134  00[illegible]056
 014136  042524  052123  004443  DH5:    .ASCIZ  /TEST#  ERRPC   ADDR    PATRN   ADR+2   GOOD    BAD/
 014144  051105  0[illegible]122  004503
 014152  042101  051104  050011
 014160  052101  047122  040411
 014166  051104  031053  043411
 014174  047517  004504  040502
 014202  000104
                                         .EVEN
 014204  001200  001116  001120  DT5:    .WORD   $TESTN,$ERRPC,$GDADR,$GDDAT,$BDADR,FILLR,$BDDAT,0
 014212  001124  001122  005060
 014220  001126  000000
 014224  040522  020115  046101  EM6:    .ASCIZ  /RAM ALTERNATE BIT PATTERN ERROR./
 014232  042524  047122  052101
 014240  020105  044502  020124
```

```
2780  014246  040520  052124  051105
2781  014254  020116  051105  047522
2782  014262  027122      000
2783  014265      124  051505  021524  DM6:    .ASCIZ  /TEST# ERRPC   ADDR    GOOD    BAD/
2784  014272  042411  051122  041520
2785  014300  040411  042104  004522
2786  014306  047507  042117  041011
2787  014314  042101      000
2788                  014320                  .EVEN
2789  014320  001200  001116  001120  DT6:    .WORD   $TESTN,$ERRPC,$GDADR,$GDDAT,$BDDAT,0
2790  014326  001124  001126  000000
2791  014334  000102  000115  054502  EM7:    .ASCIZ  /RAM BYTE TEST ERROR./
2792  014342  000104  000040  051505
2793  014350  000124  051105  047522
2794  014356  027122      000
2795  014361      124  051505  021524  DM7:    .ASCIZ  /TEST# ERRPC   ADDR    GOOD    BAD     BYTE/
2796  014376  042411  051122  041520
2797  014404  040411  042104  004522
2798  014412  047507  042117  041011
2799  014420  042101  041011  052131
2800  014426  000105
2801                  014432                  .EVEN
2802  014432  001200  001116  001120  DT7:    .WORD   $TESTN,$ERRPC,$GDADR,$GDDAT,$BDDAT,$BDADR,0
2803  014446  001124  001126  001122
2804  014454  000000
2805  014456  051120  046517  051040  EM10:   .ASCIZ  /PROM READ ERROR./
2806  014464  040105  020104  051105
2807  014472  047522  027122      000
2808  014477      124  051505  021524  DM10:   .ASCIZ  /TEST# ERRPC   ADDR    GOOD    BAD/
2809  014504  042411  051122  041520
2810  014512  040411  042104  004522
2811  014520  047507  042117  041011
2812  014526  042101      000
2813                  014532                  .EVEN
2814  014532  001200  001116  001122  DT10:   .WORD   $TESTN,$ERRPC,$BDADR,$GDDAT,$BDDAT,0
2815  014546  001124  001126  000000
2816  014554  051120  046517  051055  EM11:   .ASCIZ  /PROM-RAM COPY ERROR./
2817  014564  042501  041440  050117
2818  014572  020131  051105  047522
2819  014600  027122      000
2820  014605      124  051505  021524  DM11:   .ASCIZ  /TEST# ERRPC   PRMADD  PRMDAT  RAMADD  RAMDAT/
2821  014624  042411  051122  041520
2822  014632  041011  046522  042101
2823  014640  004504  051120  042115
2824  014646  052101  051011  046501
2825  014654  042101  004504  040522
2826  014662  042115  052101      000
2827                  014670                  .EVEN
2828  014670  001200  001116  001120  DT11:   .WORD   $TESTN,$ERRPC,$GDADR,$GDDAT,$BDADR,$BDDAT,0
2829  014704  001124  001122  001126
2830  014712  000000
2831  014714  051120  046517  053440  EM12:   .ASCIZ  /PROM WRITE-TRAP ERROR./
2832  014722  044522  042524  052055
2833  014730  040522  020120  051105
```

```
                014664 047522 027122    000
                014671    124 051505 021524 DH12:   .ASCIZ  /TEST# ERRPC  ADDR   DATWAS  DATIS/
                014676 046411 051122 042520
                014704 046411 042104 004522
                014712 040504 053524 051501
                014720 042011 052101 051511
                014726    000
                014730                       .EVEN
                014730 001200 001116 001122 DT12:   .WORD  $TESTN,$ERRPC,$BDADR,$GDDAT,$BDDAT,0
                014736 001124 001126 000000
                014744 052524 020123 044524 EM13:   .ASCIZ  /BUS TIME-OUT (TRAP) ERROR./
                014752 042515 047505 052125
                014760 040040 051124 050101
                014766 000051 051105 047522
                014774 027122    000
                015000    124 051505 021524 DH13:   .ASCIZ  /TEST#  TRAPPC  TRAPPS/
                015004 052011 040522 050120
                015012 004503 051124 050101
                015020 051520    000
                015024                       .EVEN
                015024 001200 001122 001126 DT13:   .WORD  $TESTN,$BDADR,$BDDAT,0
                015032 000000
                015034 042515 047515 054522 EM14:   .ASCII  /MEMORY MAP ERROR./<15><12>/BANK /
                015042 046440 050101 042440
                015050 051122 051117 006456
                015056 041012 047101 020113
                015064 020060    040 EM14A:  .ASCII  /0  /
                015067     055 026455 026455 EM14B:  .ASCIZ  /----------------/
                015074 026455 026455 026455
                015102 026455 026455 000055

                                     .EVEN
                015110 000000 000000        DF:     .WORD   0,0             ;OCTAL FORMAT FOR ALL.
                015114 000055               DASH:   .ASCIZ  /-/
                015116 000122               R:      .ASCIZ  /R/
                015120 000127               W:      .ASCIZ  /W/
                                            ;
                                            ;DATA BUFFER FOR RAM FAST ADDRESS TEST.
                                            ;
                015122 000400               IBUF:   .BLKW   256.

                                            ;;*****************************************************************
                016122 000240                       NOP             ;END. HERE THRU 17776 IS FREE MEMORY.
                000001                              .END
```

```
ABASE = 020000            502#    731     772
ACOM1 = 140000            503#    731     774
ACOM2 = 000004            504#    731     775
ACPUOP= 000000            731     746
ADDRS   007166           1815#   1848*   1849*   1860    1873    1880    1887*
ADOM0 = 000060            731
ADOM1 = 000000            731
ADOM10= 000000            731
ADOM11= 000000            731
ADOM12= 000000            731
ADOM13= 000000            731
ADOM14= 000000            731
ADOM15= 000000            731
ADOM2 = 000000            731
ADOM3 = 000000            731
ADOM4 = 000000            731
ADOM5 = 000000            731
ADOM6 = 000000            731
ADOM7 = 000000            731
ADOM8 = 000000            731
ADOM9 = 000000            731
ADEVCT= 000000            731     737
ADEVM = 000000            731     773
AENV  = 000000            731     742
AENVM = 000000            731     743
AFATAL= 000000            731     734
AGAIN   002624           1026#   1688
AMADR1= 000000             731     759
AMADR2= 000000             731     763
AMADR3= 000000             731     766
AMADR4= 000000             731     769
AMAPS1= 000000             731     753
AMAPS2= 000000             731     761
AMAPS3= 000000             731     764
AMAPS4= 000000             731     767
AMSCAD= 000000             731     739
AMSCLC= 000000             731     740
AMSCTY= 000000             731     733
AMTYP1= 000000             731     754
AMTYP2= 000000             731     762
AMTYP3= 000000             731     765
AMTYP4= 000000             731     768
APASS = 000000             731     736
APRIOR= 000000             731
APTCSU= 000140            2119    2627#
APTENV= 000001            2019    2112    2583    2625#
APTSIZ= 000200             880    2624#
APTSPC= 000100            2114    2585    2626#
ASHREG= 000000             731     744
ATESTM= 000000             731     735
AUNIT = 000000             731     738
AUSER = 000000             731     745
AVECT1= 000000             731     770
AVECT2= 000000             731     771
```

```
BOUMP   007276          537    1[illegible]8
BIT0  = 000001          [illegible]58    1[illegible]
BIT00 = 000001          [illegible]58
BIT01 = 000002          [illegible]48    [illegible]
BIT02 = 000004          6238    633
BIT03 = 000010          6228    632
BIT04 = 000020          [illegible]18    631
BIT05 = 000040          [illegible]38    630
BIT06 = 000100          6198    629
BIT07 = 000200          6188    628
BIT08 = 000400          6178    [illegible]7    1932
BIT09 = 001000          6168    [illegible]6    1950    2030
BIT1  = 000002          6348    1128
BIT10 = 002000          6158    2007
BIT11 = 004000          6148    1957
BIT12 = 010000          6138    1603
BIT13 = 020000          6128    1604    2014
BIT14 = 040000          6118    1605    1918
BIT15 = 100000          6108    1606
BIT2  = 000004          6338
BIT3  = 000010          6328
BIT4  = 000020          6318
BIT5  = 000040          6308
BIT6  = 000100          6298
BIT7  = 000200          6288
BIT8  = 000400          6278
BIT9  = 001000          6268
BKG     014640          1423#   1432#   1446#   1449    1457    1481
BPTVEC= 000014          642#
BYTTST  014142          1426    1430    1434    1438    1447#
CKSWR = 104410          1917    2003    2029    2668#
CR    = 000015          550#    2158    2168
CRLF  = 000200          551#    908     940     967     1000    1826    1838    1846    1855    2129    2168    2674    2685
DASH    015114          1066    2867#
DATTST  003420          1162    1165    1171#
DDISP = 177570          557#    712     868
DF      015110          794     797     800     803     806     809     812     815     818     821     824     2866#
DH1     013464          794     2706#
DH10    014457          815     2[illegible]8#
DH11    014553          818     2[illegible]0#
DH12    014671          821     2[illegible]5#
DH13    014777          824     2849#
DH2     013555          797     2718#
DH3     013651          800     2730#
DH4     014002          803     2748#
DH5     014136          806     2766#
DH6     014265          809     2783#
DH7     014361          812     2795#
DISPLA  001142          712#    [illegible]8#    876#    1972#   2006#
DISPRE  000174          523#    876
DMPGO   007552          1867    1872#   1892
DSTRT   007402          1829    1842#   1870    1897
DSWR  = 177570          556#    711     867
DT1     013516          794     2712#
```

DVMRAA.P11 CROSS REFERENCE TABLE

```
DT10    014512            815    2814#
DT11    014624            818    2828#
DT12    014730            821    2842#
DT13    015024            824    2854#
DT2     013610            797    2724#
DT3     013716            800    2738#
DT4     014050            803    2756#
DT5     014204            806    2774#
DT6     014320            809    2789#
DT7     014420            812    2802#
EMTVEC= 000030            645#    851*    852*
EM1     013436            794    2702#
EM10    014436            815    2805#
EM11    014526            818    2816#
EM12    014642            821    2831#
EM13    014744            824    2844#
EM14    015034            827    2856#
EM14A   015064           1126*   2861#
EM14B   015067           1141*   1143*   1145*   2861#
EM2     013532            797    2714#
EM3     013624            800    2726#
EM4     013734            803    2741#
EM5     014070            806    2759#
EM6     014224            809    2777#
EM7     014334            812    2791#
ERRVEC= 000004            638#    865     866*    877*   1923    1924*   1926*   1929*
FILLR   005060           1276*   1286*   1294#   1299   1313    1319    1331    1355    1365    1368    2756    2774
GNS   = ****** U            522     907     939    966     999    1825    1837    1845    1854    2659    2660    2661    2662
                          2663    2664    2666    2668    2669    2670    2671
GTSWR = 104407             902    2666#
HT    = 000011             548#   2127    2168
IBUF    015122            1240    1254    2873#
INIT1   002202             916     936#    956
INIT2   002312             953     963#    992    1024
INIT3   002460             976     982     989     996#
IOTVEC= 000020             643#    849*    850*
K1      006342            1603#   1612    1635
K2      006344            1604#   1637
K3      006346            1605#   1639
K4      006350            1606#
LF    = 000012             549#   2162    2168
LINE    007172            1817#   1839*   1840*   1877
LOC     007020            1040*   1049    1050*   1085*   1780#
MAINT   001434             838#    884    1098    1792*   1797*   1819*   1822*   1831*   1834*
MAPK    007016            1041*   1076*   1078*   1779#
MAPSW   007022             932*   1020*   1045    1055    1064    1073    1082    1088    1103    1109*   1781#   1795*
MDUMP   007124             538    1792#   1799
MEMMAP  007024            1042    1043    1105    1112    1131    1133    1777    1778    1783#
MODE    007164            1814#   1827*   1839*   1881
M1      013204             955    2674#
M2      013301            1023    2685#
M3      013412            1869    2698#
NXM     006662             951    1018    1703#   1730    1732    1865
NXM1    006674            1707#   1734
```

DVMRAA.P11 CROSS REFERENCE TABLE

```
ODUMP   007174          536  1619#
PATRN   003416         1161* 1164* 1170# 1173  1180  1189  1197
PC    =x000007          569# 8[illegible]* 951* 1018* 1100* 1162* 1165* 1186* 1278* 1288* 1336* 1426* 1430*
                       1434# 1[illegible]* 1471* 1669* 1672* 1682* 1687  1706* 1707* 1794* 1796* 1821* 1833*
                       1865* 2016* 2022* 2079* 2117* 2136* 2143* 2150* 2164* 2166* 2390* 2602* 2619*
PIRQ  = 177772          555#
PIRQVE= 000240          649#
PRM     001424          834# 923* 931  1002* 1004* 1005  1008  1489  1542  1613
PRMEND  001426          835# 926* 929* 931* 1017* 1513  1564
PRMSZ   001430          836# 924* 970* 971* 973   975*  978   981*  984*  986*  988*  1010
PRMX    001432          837# 922* 972* 980* 1581
PR0   = 000000          572#
PR1   = 000040          573#
PR2   = 000100          574#
PR3   = 000140          575#
PR4   = 000200          576#
PR5   = 000240          577#
PR6   = 000300          578#
PR7   = 000340          579#
PS    = 177776          552#  553
PSW   = 177776          553#
PTRN    005634         1425* 1429* 1437* 1444# 1464  1477
PWRVEC= 000024          644#  855*  856*  2532* 2533* 2542* 2548* 2560* 2561*
R       015116         1075  2868#
RAM     001420          832# 918* 942* 944* 945  947  1172  1179  1206  1214  1239  1253  1296
                       1304  1311  1376  1387  1398  1447  1454  1490  1504  1516  1544  1615
RAMEND  001422          833# 919* 920* 950* 1241  1297  1377
ROCHR = 104411         2449  2669#
ROLIN = 104412          968  2494  2670#
ROOCT = 104413          941  1001  1847  1856  2671#
RO1     002740         1050# 1715
RESVEC= 000010          639#
RMAP    007012         1042* 1052* 1061* 1070* 1079* 1777#
RSTRT   002620          534   933  1021  1025#
RTN1    002774         1060# 1717
RTN2    003024         1069# 1722
R0    =x000000          560# 969* 970  1049* 1050  1102* 1105* 1106  1107  1112  1114  1115  1122
                       1128  1130  1131  1132  1133  1679* 1682  2051  2052* 2053* 2060* 2061* 2062*
                       2063* 2064* 2065  2070  2075* 2077* 2081  2083  2110  2111* 2116  2121  2124*
                       2181  2191* 2195  2211  2212  2225* 2491  2495* 2496  2499  2519* 2522* 2534
                       2559* 2579  2587* 2591  2592  2594* 2595* 2596  2618* 2636  2637* 2638  2639*
                       2640* 2641* 2642*
R1    =x000001          561# 925* 927* 947* 949  1008* 1016  1122* 1123* 1124* 1125* 1126  1135*
                       1141* 1143* 1145* 1149* 1150  1171* 1174* 1177* 1183* 1205* 1209* 1213* 1218*
                       1239* 1244  1250  1253* 1255  1258  1263  1265  1306* 1307  1308  1324* 1328*
                       1333* 1342  1379* 1381  1382* 1388  1391* 1399  1401* 1411  1415  1424* 1428*
                       1433* 1436* 1455* 1459* 1461* 1467* 1476  1489* 1495  1497* 1499  1503* 1505
                       1509  1513  1515  1525  1526  1532  1542* 1544  1546* 1548  1549* 1555  1557
                       1564  1566  1612* 1618* 1620  1622* 1625  1627  1629  1631  1635* 1637* 1639*
                       1649  1703  1860* 1864  1877* 1891* 2182  2195* 2196  2200  2224* 2321  2322*
                       2325* 2331* 2492  2497* 2505* 2507* 2509* 2512* 2515  2518* 2535  2558* 2580
                       2617*
R2    =x000002          562# 948* 949* 950  1009* 1013* 1016* 1017  1172* 1173* 1179* 1180  1190
                       1191  1196* 1197* 1206* 1207* 1208  1214* 1215  1217  1225  1226  1227  1231*
```

```
SW09  = 001000            588#    598
SW1   = 000002            606#
SW10  = 002000            587#
SW11  = 004000            586#
SW12  = 010000            585#
SW13  = 020000            584#
SW14  = 040000            583#
SW15  = 100000            582#
SW2   = 000004            605#
SW3   = 000010            604#
SW4   = 000020            603#
SW5   = 000040            602#
SW6   = 000100            601#
SW7   = 000200            600#
SW8   = 000400            599#
SW9   = 001000            598#
TBITVE= 000014            640#
TIMOUT  006676            527    1715#
TKVEC = 000060            647#
TPVEC = 000064            648#
TRAPVE= 000034            646#    853#    854#
TRTVEC= 000014            641#
TST1    002630           1030#   1977
TST1A   002670           1040#   1110    1796
TST10   005536           1405    1421#   1984
TST11   006026           1440    1487#   1985
TST12   006212           1522    1538#   1986
TST13   006314           1570    1575#   1987
TST2    003362           1117    1159#   1978
TST3    003544           1166    1203#   1979
TST4    003660           1222    1237#   1980
TST5    004764           1260    1274#   1981
TST6    005022           1284#   1982
TST7    005370           1290    1374#   1983
TYPBN = 104406           1883    2664#
TYPDS = 104405           1677    2663#
TYPE  = 104401            894     937     955     964     991     997    1023    1047    1048    1057    1066    1075    1084
                         1090    1675    1678    1823    1835    1843    1852    1869    1872    1879    1895    2009    2017
                         2050    2067    2069    2072    2074    2078    2085    2132    2226    2295    2328    2357    2358
                         2361    2374    2385    2404    2453    2456    2460    2523    2525    2562    2659#
TYPOC = 104402           1886    2058    2082    2360    2660#
TYPON = 104404           2662#
TYPOS = 104403           1874    2661#
W       015120           1057    2869#
WMAP    007014           1043*   1054*   1063*   1072*   1080*   1778#
WORDS   007170           1816#   1857*   1861    1888*
WORDX   005636           1445#   1454*   1455    1464    1466*   1474    1475    1481*
WRT1    002744           1051#   1720
WRT12   006252           1555#   1725
WTRTN   006272           1559    1564#   1727
XTEST   006352           1582    1608#   1610
$APTHD  001000            671     677#
$ASTAT= ****** U         2605    2620
$ATYC   012676           2576    2578#
```

DVMRAA.P11 CROSS REFERENCE TABLE

```
$ATY1   012652          2574#
$ATY3   012660          2117    2575#
$ATY4   012670          2022    2577#
$AUTOB  001134           708#    904*   2354    2475
$BASE   001250           772#    918     919     944
$BOADR  001122           703#   1131*   1136    1147*   1191*   1192*   1225*   1263*   1343*   1351*   1352*   1361*   1362*
                        1368*   1476*   1527*   1555*   1648*   1738*   2712    2724    2738    2756    2774    2802    2814
                        2828    2842    2854
$BDDAT  001126           705#   1133*   1138    1148*   1190*   1227*   1264*   1344*   1353*   1363*   1408*   1475*   1528*
                        1557*   1650*   1739*   2712    2724    2738    2756    2774    2789    2802    2814    2828    2842
                        2854
$BELL   001164           723#   2009    2042
$BIN    011556          2324*   2327*   2328    2335#
$CDW1   001254           774#    923    1004
$CDW2   001256           775#    924     925     975
$CHARC  011026          2134*   2144*   2151    2160*   2165#
$CKSUM  011560          2346#   2668
$CNTAG  001100           691#    842     843     851     857     858     859
$CNT3   = 000000         721#
$CNTLG  012305          2357    2470#
$CNTLU  012300          2374    2469#
$CPUOP  001222           746#
$CRLF   001171           725#   1047    1048    1084    1090    1872    1895    2017    2042    2050    2069    2074    2078
                        2133    2168    2385    2469    2528
$DBLK   011246          2192    2226    2234#
$DEVCT  001204           737#
$DEVM   001252           773#
$DOAGN  006636          1671    1680    1686#
$DTBL   011236          2195    2230#
$ENDAD  006626           657     892    1682#   2037
$ENDCT  006574           857    1673#
$ENDMG  006645          1675    1690#
$ENULL  006642          1678    1689#
$ENV    001214           742#    898     915    2019    2112    2583    2607
$ENVM   001215           743#    880    2114    2119    2585
$EOP    006540          1584    1645    1663#
$EOPCT  006566           857*   1670#   1674
$ERFLG  001103           694#   1907    1946    1948    1954*   1976    2004*   2042
$ERMAX  001115           700#    860*   1948    1971*   1976
$ERROR  010214           851    2002#
$ERRPC  001116           701#   2011*   2012*   2013    2042    2056    2712    2724    2738    2756    2774    2789    2802
                        2814    2828    2842
$ERRTB  001260           792#   2064
$ERRTY  010414          2016    2049#
$ERTTL  001112           698#   2010*   2042
$ESCAP  001162           722#    859*   1970*   2033    2035    2042
$ETABL  001214           741#
$ETEND  001260           683     776#
$FATAL  001176           734#   2611*
$FFLG   013116          2574*   2577*   2605    2614*   2622#
$FILLC  001156           719#   2137    2168
$FILLS  001155           718#   2168
$GOADR  001120           702#   1265*   1266*   1349*   1359*   1409*   1410*   1474*   1525*   2738    2756    2774    2789
                        2802    2828
```

```
SGDDAT  001124            704#  1189*  1226*  1342*  1350*  1360*  1411*  1477*  1526*  1556*  1649*  2712   2724
                         2756   2774   2789   2802   2814   2828   2842
SGET42  006616           1679#
SGTSWR  011630           2358#  2666
SHD   = 000003            498    499
SHIBTS  001000            678#
SHIOCT  012472           2516*  2527#
SICNT   001104            695#  1961*  1962   1964*  1975
SILLUP  012634           2532   2548   2565#
SINTAG  001135            709#  2386   2475
SITEMB  001114            699#  2013*  2021   2042   2053
SLF     001172            726#  2042   2168   2460   2469   2528
SLFLG   013115           2615*  2621#
SLPADR  001106            696#   861*  1035*  1036   1185   1221   1279   1289   1404   1470   1521   1569   1610*
                         1644   1943*  1952*  1968*  1973   1975
SLPERR  001110            697#   862*  1036*  1178*  1185*  1212*  1221*  1279*  1289*  1303*  1386*  1404*  1453*
                         1470*  1493*  1521*  1541*  1569*  1611*  1644*  1740*  1952   1969*  1975   2032
SMODR1  001226            759#
S   R2  001232            763#
S   R3  001236            766#
S   R4  001242            769#
S   IL  001174            679     683    732#   879    898    1967   2019   2112
S   S1  001224            753#
S   2   001230            761#
S   3   001234            764#
S   4   001240            767#
SMLDR   001002            679#
SMFLG   013114           2575*  2581   2616*  2620#
S   M   012323           2361   2473#
S   AD  001210            739#  2591*  2594
S   LG  001212            740#  2596*
SMSCTY  001174            733#  2589   2597*  2609   2613*
SMSWR   012312           2358   2471#
SMTYP1  001225            754#
SMTYP2  001231            762#
SMTYP3  001235            765#
SMTYP4  001241            768#
SMXCNT  010164           1965   1975#
S.ULL   001154            717#  2139   2168
SMWTST= 000001           1027#  1156#  1200#  1234#  1271#  1281#  1371#  1418#  1484#  1535#  1572#
SOCNT   011500           2267*  2296*  2309#
SOMODE  011502           2262*  2266*  2271   2274*  2285*  2311#
SOVER   010150           1919   1944   1953   1963   1972#
SPASS   001202            736#   879*  1025*  1667*  1668*  1676   1689   1959   1976
SPASTM  001006            681#
SPOWER  012642           2563   2568#
SP   N  012474            855   2532#  2560
SP   G  012630           2563#
SP   P  012546           2542   2548#
SQ   S  001170            724#   991   2042   2168   2404   2453   2469   2525   2528
SRDCHR  012042           2417#  2669
SRDDEC= ****** U         2672
SRDLIN  012162           2445#  2670
SRDOCT  012334           2489#  2671
```

```
$DMSZ = 000010          2438#
$KTMAD  004640          1603#
$R2A  = ****** U        2672
$SAVRE= ****** U        2672
$SAVR6  012740          2541#   2549    2550#   2551#   2567#
$SCOPE  000654           849    1916#
$SETUP= 000137           840#    848     849     851     853     855     857     858     859     861     892     895     1665
                        1917    2003    2029    2037    2341    2475
$STUP = 177777           840#
$SVLAD  010114          1927    1966#
$SVPC = 000224           655#    660
$SWR  = 167400           498     499#    501#    509     510     511     512     513     514     515     721     722     723
                         858     859     861     862    1031    1160    1204    1238    1275    1285    1375    1422    1488
                        1539    1576    1660    1666    1681    1687    1689    1908    1909    1910    1911    1912    1918
                        1930    1932    1933    1946    1947    1948    1955    1956    1957    1969    1972    1975    1994
                        1995    1996    1997    1998    2007    2014    2026    2030    2042    2564
$SWREG  001216           744#    852
$SWRMK= 000000           515     516    1912    1913    1936
$SWSBT  010166          1942    1976#
$TESTN  001200           735#   1033*   1967*   2712    2724    2738    2756    2774    2789    2802    2814    2828    2842
                        2854
$TIMES  001160           721#    858*   1031*   1160*   1204*   1238*   1275*   1285*   1375*   1422*   1488*   1539*   1576*
                        1583*   1666*   1955*   1962    1965*   1975
$TKB    001146           714#   2339    2350    2367    2421    2427
$TKS    001144           713#   2339    2348    2364    2378*   2419    2425
$TN   = 000014           498#   1027    1031#   1117    1156    1160#   1166    1200    1204#   1222    1234    1238#   1260
                        1271    1275#   1281    1285#   1290    1371    1375#   1405    1418    1422#   1440    1484    1488#
                        1522    1535    1539#   1570    1572    1576#   1937    1977
$TPB    001152           716#   2157*   2168
$TPFLG  001157           720#   2106    2168
$TPS    001150           715#   2155    2168
$TRAP   013120           853    2636#
$TRAP2  013142          2647#   2658
$TRP  = 000014          2651#   2660#   2661#   2662#   2663#   2664#   2665#   2666    2667#   2668    2669#   2670#   2671#
                        2672#
$TRPAD  013154          2641    2658#
$TSTM   001004           680#
$TSTNM  001102           693#   1032*   1665*   1907    1939*   1966*   1967    1972    1976    2006    2042
$TTYIN  012270          2446    2447    2464    2468#
$TYPCH  011504          2321#   2664
$TYPUS  011032          2180#   2663
$TYPE   010550          2106#   2602    2651    2659
$TYPEC  010762          2136    2143    2150    2155#   2156    2390
$TYPEX  011030          2161    2163    2166#
$TYPOC  011302          2265#   2660
$TYPON  011316          2264    2267#   2662
$TYPOS  011256          2260#   2661
$UNIT   001206           738#
$UNITM  001010           682#
$USWR   001220           745#
$VECT1  001244           770#
$VECT2  001246           771#
$XTSTR  007666          1921#
$$GET4= 000000          1681#
```

```
$$SWOB= 000014          1976#  1977   1978#  1979#  1980#  1981#  1982#  1983#  1984#  1985#  1986#  1987#  1988#
$OFILL  011501          2261*  2265*  2275   2310#
$40CAT= ****** U        1918   2016
.       = 016124         518#   522#   526#   529#   532#   655    656#   658#   660#   661#   667    668#   670#
                         672#   690#   727    846    861    862    885   1000#  1099   1129   1689   1693   1783#
                        1785#  1838#  1855#  1975   1976   2042   2088#  2168   2234#  2339   2468#  2469   2475
                        2528   2544   2566   2623#  2723#  2737#  2788#  2813#  2827#  2841#  2853#  2873#
.$ASTA= ****** U        2575   2578
.$X   = 001000           667#   672
```

CROSS REFERENCE TABLE

```
COMMEN    650#
ENDCOM    650#
ERROR     544#  1152  1193  1228  1267  1345  1354  1364  1412  1478  1529  1558  1651  1741
ESCAPE    650#
GETPRI    650#
GETSWR    650#  895#
MULT      650#
NEWTST    650#  1027  1156  1200  1234  1271  1281  1371  1418  1484  1535  1572
POP       650#   942  1002  1737  1739  1848  1857  2221  2617  2553  2554  2617  2618
PUSH      650#  1873  1880  2180  2491  2534  2540  2578  2580  2601
REPORT    650#
SCOPE     545#  1159  1203  1237  1274  1284  1374  1421  1487  1538  1575  1664
SETPRI    650#
SETTRA   2651#  2660  2661  2662  2663  2664  2666  2668  2669  2670  2671
SETUP     650#   840
SKIP      650#  1117  1166  1222  1260  1290  1405  1440  1522  1570  1645
SLASH     650#
SPACE     650#
STARS     650#   653   664   666   673   686   727   730  1027  1029  1118  1154  1156  1158  1188
         1199  1200  1202  1224  1233  1234  1236  1262  1270  1271  1273  1281  1283  1338  1370
         1371  1373  1407  1417  1418  1420  1473  1483  1484  1486  1524  1534  1535  1537  1552
         1561  1572  1574  1647  1654  1657  1694  1710  1746  1788  1800  1904  1990  2044  2091
         2170  2237  2314  2338  2341  2409  2438  2477  2530  2546  2573  2630  2673  2875
SWRSU     650#  863#
TRMTRP   2651#
TYPBIN    650#
TYPDEC    650#  1676
TYPNAM    650#   888
TYPNUM    650#
TYPOCS    650#
TYPOCT    650#  2056  2080  2359
TYPTXT    650#   936   963   996  1823  1835  1842  1851
$$CMRE    684#
$$CMTM    684#
$$ESCA    650#
$$NEWT    650#  1027  1156  1200  1234  1271  1281  1371  1418  1484  1535  1572
$$SET    2651#  2660  2661  2662  2663  2664  2666  2668  2669  2670  2671
$$SETM    879#
$$SKIP    650#  1117  1166  1222  1260  1290  1405  1440  1522  1570
.EQUAT    487#   540
.HEADE    487#   488
.SETTR    487#
.SETUP    487#   840
.SWRHI    487#   505
.SWRLO    516#
.TRMTR    487#
.$ACT1    487#   651
.$APTB    728#
.$APTH    487#   662
.$APTY    487#  2571
.$CATC    487#   516
.$CMTA    487#   684
.$EOP     487#  1655
.$ERRO    487#  1988
```

DVMRAA.P11 CROSS REFERENCE TABLE

DVMRAA.P11 CROSS REFERENCE TABLE

```
EMT      544
HALT     572  1744 1798 2028 2039 2108 2543 2565
INC      840  1149 1622 1667 1961 2010 2198 2291 2299 2401 2551 2613
INCB    1966  2004 2164
IOT      545
JMP      533   534  536  537  538  933 1021 1024 1584 1687 1745 1799
JSR      951  1018 1162 1165 1278 1288 1426 1430 1434 1438 1682 1794 1796 1821 1833
        1865  2016 2022 2117 2136 2143 2150 2220 2222
MOV      843   847  849  850  851  852  853  854  855  856  857  861  862  865  866
         867   868  873  875  876  877  878  879  919  923  924  925  926  932  942
         944   947  948  950  958  959  960  961 1003 1004 1008 1009 1010 1017 1026
        1031  1033 1035 1036 1041 1042 1043 1044 1045 1078 1105 1109 1122 1131 1133
        1160  1164 1171 1172 1173 1174 1175 1176 1177 1189 1190 1191 1197 1204 1205
        1206  1207 1212 1213 1214 1215 1216 1217 1218 1231 1232 1239 1240 1241 1244
        1250  1253 1254 1263 1264 1265 1266 1267 1268 1277 1279 1296 1297 1289 1296 1297
        1299  1303 1304 1306 1307 1308 1309 1310 1311 1343 1344 1349 1350 1351 1353 1355
        1359  1360 1361 1363 1365 1366 1367 1368 1369 1377 1379 1381 1385 1387 1398 1404
        1408  1409 1411 1415 1422 1423 1424 1425 1426 1446 1447 1448 1449 1453 1454 1456
        1470  1474 1475 1476 1477 1481 1482 1483 1484 1490 1491 1492 1493 1499 1504 1516
        1517  1518 1521 1525 1526 1527 1528 1532 1538 1539 1541 1542 1546 1549 1555 1556
        1557  1569 1576 1610 1611 1612 1613 1644 1646 1648 1649 1650 1672 1676 1679 1717
        1722  1727 1734 1738 1739 1740 1792 1793 1818 1819 1820 1828 1831 1832 1838 1840
        1848  1857 1860 1861 1873 1877 1880 1923 1924 1926 1929 1939 1943 1952 1964
        1965  1968 1969 1972 1973 2006 2011 2032 2035 2051 2056 2065 2070 2075 2077
        2081  2110 2111 2116 2124 2139 2181 2182 2183 2184 2185 2186 2187 2192 2195
        2215  2221 2222 2223 2224 2225 2227 2228 2260 2268 2269 2270 2276 2283 2301
        2302  2303 2304 2305 2321 2322 2331 2332 2333 2359 2383 2388 2417 2418 2445
        2446  2461 2462 2463 2464 2489 2490 2491 2492 2493 2496 2498 2515 2516 2517
        2518  2519 2532 2533 2534 2535 2536 2537 2538 2539 2540 2541 2542 2548 2549
        2553  2554 2555 2556 2557 2558 2559 2560 2561 2579 2580 2587 2591 2596 2597
        2599  2601 2611 2617 2618 2636 2637 2641 2647 2648
MOVB     860   904  970 1032 1126 1141 1143 1145 1459 1935 1967 1971 2013 2021 2121
        2149  2157 2190 2193 2207 2210 2219 2261 2262 2265 2266 2267 2271 2274 2275
        2294  2324 2350 2367 2421 2427 2450 2455 2499 2574 2575 2577 2639
NEG     2189  2272
NOP     1030  1463 1608 1609 1642 1643 1683 1684 1685 1896 2876
RESET   1681
ROL     1052  1054 1061 1063 1070 1072 1147 1148 1324 1328 2278 2280 2281 2282 2284
        2325  2506 2508 2510
RTI      874  1718 1723 1728 1735 1974 2041 2126 2229 2306 2334 2389 2437 2465 2520
        2564  2649
RTS      886  1100 1186 1336 1471 1706 1707 2079 2166 2619 2642
SEC     1051  1053 1069 1327 2323
SUB      984   986 1503 2012 2196 2594
TRAP    2651  2660 2661 2662 2663 2664 2666 2668 2669 2670 2671
TST      894   896  913  973 1098 1106 1114 1130 1132 1136 1138 1208 1217 1322 1457
        1509  1510 1515 1566 1581 1623 1703 1704 1881 1925 1945 1959 2026 2033 2083
        2123  2131 2153 2201 2211 2289 2381 2396 2514 2521 2589 2607 2609 2638
TSTB    1045  1055 1064 1073 1082 1088 1103 1946 2106 2155 2203 2217 2348 2364 2419
        2425  2581 2592 2605
.ASCII   724   725 2674 2685 2856 2860
.ASCIZ   723   726  908  940  967 1000 1690 1826 1838 1846 1855 2087 2469 2470 2471
        2473  2568 2679 2690 2698 2702 2706 2714 2718 2726 2730 2741 2748 2759 2766
        2777  2783 2791 2795 2805 2808 2816 2820 2831 2635 2844 2849 2861 2867 2868
```

DVMRAA.P11 CROSS REFERENCE TABLE

```
            2969
.BLKB       2973
.BLKW       1783  1785  2234  2873
.BYTE        693   694   699   700   708   709   717   718   719   720   742   743   753   754   761
             762   764   765   767   768  1689  1781  1875  1876  1899  2023  2024  2307  2308  2309
            2310  2335  2466  2467  2620  2621  2622
.DSPRL      1155  1571  2406
.EMUL        487  1034  1540  2339
.END        2877
.ENDC        493   512   514   515   516   525   544   626   650   654   658   660   665   667   674
             687   691   693   721   722   723   724   729   731   753   761   764   767   770   771
             772   773   774   775   778   840   847   848   851   853   855   857   858   859   861
             863   864   892   894   900   906   908   940   967  1000  1028  1029  1030  1031  1032
            1118  1119  1155  1157  1158  1159  1160  1161  1167  1189  1200  1201  1202  1203  1204
            1205  1223  1225  1234  1235  1236  1237  1238  1239  1261  1263  1271  1272  1273  1274
            1275  1276  1282  1283  1284  1285  1286  1291  1339  1371  1372  1373  1374  1375  1376
            1406  1408  1418  1419  1420  1421  1422  1423  1441  1474  1484  1485  1486  1487  1493
            1499  1523  1525  1535  1536  1537  1538  1539  1540  1553  1562  1571  1573  1574  1575
            1576  1577  1646  1648  1655  1658  1659  1660  1662  1665  1671  1674  1675  1679  1681
            1687  1689  1690  1693  1695  1711  1747  1789  1801  1826  1838  1846  1855  1905  1908
            1913  1918  1920  1931  1934  1946  1948  1950  1957  1961  1966  1968  1972  1975  1976
            1993  1991  1994  2004  2011  2016  2017  2018  2028  2037  2041  2042  2045  2060  2089
            2092  2121  2171  2238  2315  2339  2340  2342  2370  2406  2410  2438  2439  2446  2448
            2451  2453  2469  2475  2478  2484  2520  2521  2540  2541  2547  2553  2554  2564  2571
            2574  2575  2578  2605  2620  2631  2639  2640  2659  2660  2661  2662  2663  2664  2665
            2666  2667  2668  2669  2670  2671  2672  2674  2676
.EQUIV       544   545   553   598   599   600   601   602   603   604   605   606   607   626   627
             628   629   630   631   632   633   634   635   636
.EVEN        731   908   940   967  1000  1826  1827  1828  1855  2088  2570  2623  2711  2723  2737
            2755  2773  2788  2801  2813  2827  2841  2842  2865
.IF          489   512   513   514   515   516   520   521   608   636   653   656   658   664   666
             673   686   690   692   721   722   723   724   728   730   753   761   764   767   770
             771   772   773   774   775   776   777   846   847   849   851   853   855   857
             858   859   861   879   891   892   893   894   907   939   966   999  1027  1029
            1031  1032  1117  1118  1154  1156  1158  1160  1161  1166  1188  1199  1200  1202  1204
            1205  1222  1224  1233  1234  1236  1238  1239  1260  1262  1270  1271  1273  1275  1276
            1281  1283  1285  1286  1290  1338  1370  1371  1373  1375  1376  1405  1407  1417  1418
            1420  1422  1423  1440  1473  1483  1484  1486  1488  1489  1522  1524  1534  1535  1537
            1539  1540  1552  1561  1570  1572  1574  1576  1577  1645  1647  1654  1657  1658  1659
            1660  1661  1662  1664  1670  1673  1675  1679  1681  1687  1689  1690  1694  1710  1746
            1788  1810  1825  1837  1845  1854  1904  1907  1912  1918  1930  1932  1933  1934  1946
            1947  1948  1957  1959  1967  1969  1974  1975  1976  1990  1993  2004  2007  2014  2016
            2017  2019  2026  2030  2037  2041  2042  2044  2059  2075  2091  2112  2170  2237  2314
            2338  2340  2341  2342  2370  2409  2410  2438  2446  2447  2451  2452  2468  2469  2475
            2477  2480  2496  2530  2540  2541  2546  2553  2554  2562  2564  2568  2573  2575  2578
            2605  2620  2630  2636  2640  2651  2660  2661  2662  2663  2664  2666  2668  2669  2670
            2671  2672  2673  2875
.IFF         512   514   515   516   542   654   658   660   665   667   674   687   690   693   721
             728   731   847   891   893  1027  1028  1029  1030  1031  1118  1119  1155  1157  1158
            1159  1160  1167  1189  1200  1201  1202  1203  1204  1223  1225  1234  1235  1236  1237
            1238  1261  1263  1271  1272  1273  1274  1275  1282  1283  1284  1285  1291  1339  1371
            1372  1373  1374  1375  1406  1408  1418  1419  1420  1421  1422  1441  1474  1484  1485
            1486  1487  1488  1523  1525  1535  1536  1537  1538  1539  1553  1562  1571  1573  1574
            1575  1576  1645  1648  1655  1658  1661  1665  1671  1674  1689  1695  1711  1747  1789
```

```
                1801  1905  1931  1934  1946  1948  1975  1976  1991  1993  2007  2037  2042  2045  2060
                2069  2072  2171  2238  2315  2339  2342  2410  2412  2417  2438  2439  2448  2452  2469
                2478  2531  2547  2564  2574  2631  2637  2674  2876
.IFT             908   940   967  1000  1826  1838  1846  1855  1956  2017  2412  2417  2501  2521  2528
.IFTF            908   940   967  1000  1826  1838  1846  1855  1954  2016  2357  2410  2413  2497  2505
                2527
.IIF             488   493   498   499   509   510   511   512   515   516   522   727   731   848   851
                 857   858   859   861   862   892  1659  1665  1666  1677  1689  1693  1908  1909  1910
                1911  1912  1913  1917  1936  1955  1956  1972  1975  1976  1994  1995  1996  1997  1998
                2003  2029  2037  2042  2057  2082  2168  2339  2360  2461  2469  2475  2528  2659  2660
                2661  2662  2663  2664  2666  2668  2669  2670  2671
.IRP             840   942  1002  1027  1156  1200  1234  1271  1281  1371  1418  1484  1535  1572  1739
                1739  1848  1857  1873  1880  1937  1977  1978  1979  1990  1981  1982  1983  1984  1985
                1986  1987  2181  2221  2491  2517  2534  2540  2553  2554  2579  2580  2601  2617  2618
.LIST              1   487   515   522   650   721   728   731   840   863   892   895   908   940   967
                1000  1027  1031  1156  1160  1200  1204  1234  1238  1250  1271  1275  1281  1285  1371
                1375  1418  1422  1484  1488  1535  1539  1572  1576  1665  1681  1826  1838  1846  1855
                1912  1976  1978  1979  1980  1981  1982  1983  1984  1985  1986  1987  1988  2037  2438
                2651  2659  2660  2661  2662  2663  2664  2665  2666  2667  2668  2669  2670  2671  2672
.MACRO           516   684   879  2651
.MCALL           487   650   728   863   895
.MEXIT           777
.NLIST             1   487   515   522   650   721   728   731   840   863   892   895   908   940   967
                1000  1027  1031  1156  1160  1200  1204  1234  1238  1250  1271  1275  1281  1285  1371
                1375  1418  1422  1484  1488  1535  1539  1572  1576  1665  1681  1826  1838  1846  1855
                1912  1976  1978  1979  1980  1981  1982  1983  1984  1985  1986  1987  1988  2037  2438
                2651  2659  2660  2661  2662  2663  2664  2665  2666  2667  2668  2669  2670  2671  2672
.PAGE            516   684   778   831   884   934  1025  1076  1118  1156  1200  1234  1271  1371  1418
                1484  1535  1572  1655  1693  1746  1800  1901  1988  2042  2089  2168  2235  2312  2336
                2475  2528  2571  2628  2672
.REM               1
.REPT            522  1250  1977
.SBTTL           505   516   540   651   662   684   728   778   831   841   888   895   909  1027  1156
                1200  1234  1271  1281  1371  1418  1484  1535  1572  1655  1693  1901  1902  1988  2042
                2089  2168  2235  2312  2336  2475  2528  2571  2628  2651  2672
.TITLE           488
.WORD            522   523   524   527   530   659   678   679   680   681   682   683   692   695   696
                 697   698   701   702   703   704   705   706   707   710   711   712   733   734   735
                 736   737   738   739   740   744   745   746   759   763   766   769   770   771   772
                 773   774   775  1670  1673  1688  1777  1778  1779  1780  1977  1978  1979  1980  1981
                1982  1983  1984  1985  1986  1987  2068  2073  2118  2165  2311  2524  2527  2563  2603
                2658  2712  2724  2738  2756  2774  2789  2802  2814  2828  2842  2854  2866
```

ERRORS DETECTED: 0

*DVMRAA.BIC,DVMRAA.SEQ/SOL/NL:TOC=DVMRAA.P11
RUN-TIME: 26 10 1 SECONDS
CORE USED: 26K

EOF DVMRAA SEQ 00010000 770526 PDP10 411