IDENTIFICATION

PRODUCT CODE: AC-S435A-MC

PRODUCT NAME: CVMSAA0 0-2 MEG MEM EXER

PRODUCT DATE: APRIL,1981

MAINTAINER: DIAGNOSTIC ENGINEERING

REVISION HISTORY

REVISION A: APRIL 1981

TABLE OF CONTENTS

1.0 GENERAL PROGRAM INFORMATION.

1.1 PROGRAM PURPOSE (ABSTRACT)

THIS PROGRAM HAS THE ABILITY TO TEST MEMORY FROM ADDRESS 000000 TO ADDRESS 17757777. IT DOES SO USING:

A. UNIQUE ADDRESSING TECHNIQUES
B. WORSE CASE NOISE PATTERNS, AND
C. INSTRUCTION EXECUTION THRUOUT MEMORY.

THE INTENT OF THIS PROGRAM IS TO TEST AS COMPREHENSIVELY AS POSSIBLE ALL MOS MEMORIES USED ON THE LSI-11 BUS WITHOUT CONCENTRATING ON ANY ONE SYSTEM. ALTHOUGH THE TESTS RELATE TO GENERAL DESIGNS THEY MAY BE COMPLETE FOR CERTAIN SYSTEMS. THIS TEST IS ALSO NOT INTENDED TO BE A TOTAL 100% TEST OF THE MEMORY. OTHER TESTS THAT DO I/O MAY FIND MEMORY PROBLEMS THAT THIS TEST IS UNABLE TO.

1.2 SYSTEM REQUIREMENTS

A. HARDWARE REQUIREMENTS

LSI-11/2,LSI-11/23 BUS FAMILY PROCESSORS.
MINIMUM OF 16KW OF MEMORY.

OPTIONAL...
ANY PARITY MEMORY CONTROL MODULE.
KTF11 MEMORY MANAGEMENT.

B. SOFTWARE REQUIREMENTS

THE SMALLEST UNIT OF MEMORY THIS PROGRAM WILL RECOGNIZE IS 8K. IF ANY ADDRESS IN A 8K BANK CAUSES A TIME OUT TRAP, THAT ENTIRE BANK OF MEMORY IS IGNORED BY THE PROGRAM. PROGRAM TESTS IN 8KW BANKS,UNLESS IT IS THE LAST 4KW BEFORE THE I/O PAGE OR LAST 6KW IF 30KW SYSTEM.

THE PROGRAM IS DESIGNED TO EXERCISE THE VECTOR PORTION OF MEMORY (LOCATIONS 0-776) IN EXACTLY THE SAME MANNER AS THE REST OF MEMORY. TO MAKE THIS POSSIBLE, WITHOUT REQUIRING MEMORY MANAGEMENT, NO SOFTWARE TRAPS ARE USED IN THE PROGRAM. THIS MEANS THAT IF MEMORY MANAGEMENT IS NOT AVAILABLE OR IS DISABLED (SW12=1), IF THE PROGRAM IS RELOCATED OUT OF BANK 0, IF LOCATION 0-776 ARE SELECTED FOR TEST, AND IF AN UNEXPECTED HARDWARE TRAP OCCURS, THE RESULTS WILL BE UNPREDICTABLE.

THE PROGRAM HAS THE PROPER INTERFACE CODE TO ALLOW RUNNING UNDER THE AUTOMATED MANUFACTURING TEST LINE SYSTEM - ACT11 AND APT.

1.2 HARDWARE RESTRICTIONS

IF THE SYSTEM HAS MIXED MSV11-D (18 BIT) MEMORIES WITH MSV11-L OR MSV11-P (22 BIT) MEMORIES,IT IS RECOMMENDED TO SET UP THE MSV11-L OR MSV11-P TO 18 BIT MODE. AS THE PROGRAM AUTOSIZES FOR NONCONTIGUOUS MEMORY, IT IS POSSIBLE TO RECEIVE RESPONSES FROM THE "MSV11-D" MEMORY WHEN SIZING ABOVE 18 BITS. THIS CAN CAUSE INCORRECT MEMORY SIZING AND FALSE ERRORS.

1.3 RELATED DOCUMENTS AND STANDARDS

A. PROGRAMMING PRACTICES - DOCUMENT NO. 175-003-009-01
B. PDP-11 MAINDEC SYSMAC PACKAGE - MAINDEC-11-DZQAC-C5-D
C. THE APPLICABLE MEMORY SYSTEM MAINTENANCE MANUAL
D. THE APPLICABLE CIRCUIT SCHEMATICS

1.4 DIAGNOSTIC HIERARCHY PREREQUISITES

BEFORE RUNNING THIS PROGRAM, A CPU DIAGNOSTIC SHOULD BE RUN TO VERIFY THE FUNCTIONALITY OF THE PROCESSOR AND PDP-11 INSTRUCTION SET. FOR LSI-11/23:CJKDB_ DIAG
FOR LSI-11/2 :CVKAA_ DIAG

IF MEMORY MANAGEMENT IS TO BE USED, THEN THE KTF11 DIAGNOSTIC CJKDA_ SHOULD ALSO BE RUN BEFORE THIS PROGRAM.

1.5 ASSUMPTIONS

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

2.0 OPERATING INSTRUCTIONS

2.1 LOADING AND STARTING PROCEDURES

2.1.1 LOAD THE PROGRAM USING XXDP OR ANY STANDARD ABSOLUTE LOADER.

2.1.2 STARTING ADDRESS 200:

NORMAL PROGRAM EXECUTION.

2.1.3 STARTING ADDRESS 204:

RESTART PROGRAM USING PREVIOUSLY SELECTED PARAMETERS.

2.2 SPECIAL ENVIRONMENTS

IF THE PROGRAM IS RUN IN AUTOMATIC MODE UNDER ACT11 OR APT11 THE PROGRAM IS DONE AFTER THE FIRST PASS. ALSO, THE PROGRAM DOES NOT RELOCATE TO TEST THE LOWER 8K OF MEMORY, AFTER THE FIRST PASS.

2.3 PROGRAM OPTIONS

THE SOFTWARE SWITCH REGISTER (LOC. 176) IS USED FOR ALL OPERATIONAL SWITCH SETTINGS.
THE USER CAN TYPE CTRL G (^G) TO ALLOW SWR CHANGES DURING PROGRAM EXECUTION.

SW15 = 1 OR UP....	HALT ON ERROR
SW14 = 1 OR UP....	LOOP ON TEST
SW13 = 1 OR UP....	INHIBIT ERROR TYPEOUT
SW12 = 1 OR UP....	INHIBIT MEMORY MANAGEMENT (INITIAL START ONLY)
SW11 = 1 OR UP....	INHIBIT SUBTEST ITERATION (NOT USED)
SW10 = 1 OR UP....	RING BELL ON ERROR
SW9 = 1 OR UP....	LOOP ON ERROR
SW8 = 1 OR UP....	LOOP ON TEST IN SWR<4:0>
SW7 = 1 OR UP....	INHIBIT PROGRAM RELOCATION
SW6 = 1 OR UP....	INHIBIT PARITY ERROR DETECTION
SW5 = 1 OR UP....	INHIBIT EXERCISING VECTOR AREA (LOCATIONS 0-1000).

2.4 EXECUTION TIMES

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

A. FOR NON-PARITY MEMORY

FULL PASS:	< 5 MINUTES FOR 64KW
	30 MINUTES FOR 1280 KW

B. FOR PARITY MEMORY

FULL PASS:	5 MINUTES FOR 64KW
	20 MINUTES FOR 256KW
	117 MINUTES FOR 1280KW

APT TIME IS SETUP FOR 256KW (APPROX. 20 MIN).

*PLEASE MAKE CHANGES TO APT TIMES AS MEMORY INCREASES OR DECREASES IN SIZE.

3. ERROR INFORMATION

3.1 ERROR REPORTING

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

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

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

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

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

MA = MEMORY ADDRESS

REG = PARITY REGISTER ADDRESS.

PS = PROCESSOR STATUS WORD.

IUT = INSTRUCTION UNDER TEST.

S/B = WHAT CONTENTS SHOULD BE.

WAS = WHAT CONTENTS WAS.

3.2 ERROR HALTS

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

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

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

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

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

E. IN THE MEMORY MAPPING ROUTINE OR ANY OF THE ADDRESS CONTROL ROUTINES, FAILURES TO FIND A MEANINGFUL MAP.

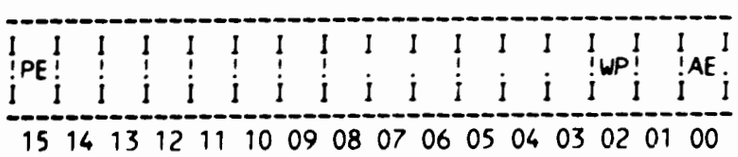
4.0 PERFORMANCE AND PROGRESS REPORTS

NOT APPLICABLE

5.0 DEVICE INFORMATION TABLES

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

5.2 MOS PARITY REGISTER



BIT ASSIGNMENTS ARE DEFINED AS FOLLOWS:

BIT15 PARITY ERROR

BIT02 WRITE WRONG PARITY .NORMAL PARITY (ODD) WHEN CLEAR; OTHER PARITY (EVEN) WHEN SET

BIT00 ACTION ENABLE NO ACTION WHEN CLEAR. TRAP TO VECTOR 114 WHEN SET.

BITS 5-11 ADDRESS BITS FOR A11-A17 OR ADDRESS BITS 18-21,IF BIT 14=1 LOCATES PARITY ERROR TO A 1K SEGMENT OF MEMORY.

BIT 14...ALWAYS READ AS A ZERO ON 128KW. IN A 2044 MACHINE,R/W AND IS THE EXTENDED CSR READ ENABLE BIT FOR A18-A21 IN BITS 5-8.

6.0 SUB-TEST SUMMARIES

6.1 SECTION 1: ADDRESS TESTS.

THESE TESTS VERIFY THE UNIQUENESS OF EVERY MEMORY ADDRESS.

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

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

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

TEST 4 WRITES THE 8K BANK NUMBER INTO EACH BYTE OF THAT BANK AND CHECKS IT.

TEST 5 WRITES THE COMPLEMENT OF THE BANK NUMBER INTO EACH BYTE OF THAT BANK AND CHECKS IT.

6.2 SECTION 2: WORST CASE NOISE TESTS.

THESE ARE INTENDED TO APPLY MAXIMUM STRESS TO THE VARIOUS TYPES OF PDP-11 MOS MEMORIES.

TEST 6 AND TEST 7 ARE SUPPLIED TO ALLOW THE OPERATOR TO SELECT A SINGLE WORD DATA PATTERN (SA-204) AND SCOPE ON EITHER THE WRITING (DATO) IN TEST 6 OR THE READING (DATI) IN TEST 7 OF THAT DATA.
LOCATION .CONST:0 SHOULD BE CHANGED IF A DIFFERENT SINGLE WORD DATA PATTERN IS CONSIDERED.

TEST 10 WRITES AND THEN CHECKS A SERIES OF SINGLE WORD PATTERNS WHICH ARE DESIGNED TO STRESS PARITY MEMORY.

TEST 11 WRITES ALL MEMORY WITH 1'S IN EVERY BIT AND THEN "RIPPLES" A "0" THROUGH IT.

TEST 12 WRITES ALL MEMORY WITH 0'S IN EVERY BIT AND THEN "RIPPLES" A "1" THROUGH IT.

TEST 13 WRITES WRONG PARITY IN EACH BYTE OF MEMORY AND CHECKS THAT THE PARITY DETECTION LOGIC WORKS. THIS TEST IS SKIPPED FOR NON-PARITY MEMORY.

TEST 14 WRITE "RANDOM" PROGRAM CODE THROUGH MEMORY AND CHECKS IT.

6.3 SECTION 3: INSTRUCTION EXECUTION TESTS.

THIS GROUP OF TESTS PLACE INSTRUCTIONS IN THE MEMORY UNDER TEST, THEN EXECUTES THE INSTRUCTIONS, AND FINALLY, CHECKS

THAT THEY EXECUTED CORRECTLY.

TEST 15 EXECUTES AN INSTRUCTION WHICH DOES A DATI AND A DATO ON THE MEMORY UNDER TEST.

TEST 16 EXECUTES AN INSTRUCTION WHICH DOES A DATI AND A DATOB ON THE LOW BYTE OF MEMORY UNDER TEST.

TEST 17 EXECUTES AN INSTRUCTION WHICH DOES A DATI AND A DATOB ON THE HIGH BYTE.

TEST 20 EXECUTES AN INSTRUCTION WHICH DOES A DATIO AND A DATO.

TEST 21 EXECUTES AN INSTRUCTION WHICH DOES A DATIO AND A DATOB ON THE LOW BYTE.

TEST 22 EXECUTES AN INSTRUCTION WHICH DOES A DATIO AND A DATOB ON THE HIGH BYTE.

6.4 SECTION 4: MOS TESTS

TEST 23 -WRITES A PATTERN OF 000377 THROUGH MEMORY, THEN COMPLEMENTS IT ADDRESSING DOWNWARD, COMPLIMENTS THE NEW PATTERN ADDRESSING UPWARD, COMPLIMENTS THE THIRD PATTERN ADDRESSING UPWARD AND FINALLY COMPLIMENTS THIS NEW AB PATTERNS ADDRESSING DOWNWARD.

TEST 24-25 WRITE A CHECKERBOARD THROUGH MEMORY THEN STALLS FOR 2 SECONDS AND THEN VERIFIES NO DATA HAS CHANGED.

6.5 SPECIAL TOGGLE IN TESTS

6.5.1 TOGGLE-IN-PROGRAM #1

THE FOLLOWING IS A TOGGLE IN MEMORY ADDRESS TEST. THIS TEST IS USEFUL WHEN AN ADDRESS SELECTION FAILURE IS SUSPECTED INVOLVING THE FIRST 8K OF MEMORY. THIS PROGRAM WRITES THE VALUE OF EACH ADDRESS INTO ITSELF STARTING WITH THE LOWER LIMIT (200) AND CONTINUING TO THE UPPER LIMIT. AFTER ALL ADDRESSES HAVE BEEN WRITTEN EACH ADDRESS IS CHECKED FOR THE CORRECT CONTENTS STARTING WITH THE UPPER LIMIT AND CONTINUING TO THE LOWER LIMIT.

```
LOCATION CONTENTS       MNEMONIC     COMMENT
  10     012700         MOV #200,R0  ;GET FIRST ADDRESS
  12     000200                      ;TO TEST
                                     ;(EXAMPLE START ADDRESS)
  14     010001         MOV R0,R1    ;SAVE IN R1
  16     020037     1$: CMP R0,@#SWR ;CHECK UPPER LIMIT
  20     000176                      ;(IN SOFTWARE SWITCH REGISTER)
  22     001403         BEQ 2$       ;BRANCH IF AT UPPER LIMIT
  24     010010         MOV R0,(R0)  ;LOAD VALUE INTO ADDRESS
  26     005720         TST (R0)+    ;STEP TO NEXT ADDRESS
  30     000772         BR 1$        ;LOOP UNTIL DONE
  32     010004     2$: MOV R0,R4    ;SAVE UPPER LIMIT
  34     020001     3$: CMP R0,R1    ;CHECK IF AT LOWER LIMIT
```

```
 • 36     001767        BEQ 1$         ;BRANCH IF DONE
   40     024000        CMP -(R0),R0   ;CHECK DATA WRITTEN
   42     001774        BEQ 3$         ;BRANCH IF OK
   44     000000        HALT           ;ERROR
   46     000772        BR 3$          ;LOOP BACK
```

6.5.2 TOGGLE-IN-PROGRAM #2

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

THIS IS THE PATCH TO TOGGLE-IN-PROGRAM #1:

```
LOCATION CONTENTS       MNEMONIC     COMMENT
  36     001404         BEQ 4$       ;BRANCH TO PROGRAM #2
```

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

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

7.0 PROGRAM LISTING
ATTACHED

```
                        .TITLE  CVMSAA  0-2 MEGAWORD MEMORY EXERCISER, 16K VER
                        ;*COPYRIGHT (C) 1981
                        ;*DIGITAL EQUIPMENT CORP.
                        ;*MAYNARD, MASS. 01754
                        ;*
                        ;*PROGRAM BY SAM/BARRY/BILL
                        ;*
                        ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
                        ;*PACKAGE (MAINDEC-11-DZQAC-C5), JAN, 1981.
                        ;*

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

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

                                                ;*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  MEMORY MANAGEMENT DEFINITIONS

                                    ;*KT11 VECTOR ADDRESS

                    000250          MMVEC=  250

                                    ;*KT11 STATUS REGISTER ADDRESSES

                    177572          SR0=    177572
                    177574          SR1=    177574
                    177576          SR2=    177576
                    172516          SR3=    172516

                                    ;*KERNEL ''I'' PAGE DESCRIPTOR REGISTERS

                    172300          KIPDR0= 172300
                    172302          KIPDR1= 172302
                    172304          KIPDR2= 172304
                    172306          KIPDR3= 172306
                    172310          KIPDR4= 172310
                    172312          KIPDR5= 172312
                    172314          KIPDR6= 172314
                    172316          KIPDR7= 172316

                                    ;*KERNEL ''I'' PAGE ADDRESS REGISTERS

                    172340          KIPAR0= 172340
                    172342          KIPAR1= 172342
                    172344          KIPAR2= 172344
```

```
                        172346          KIPAR3= 172346
                        172350          KIPAR4= 172350
                        172352          KIPAR5= 172352
                        172354          KIPAR6= 172354
                        172356          KIPAR7= 172356

                        000000          UP = 0                  ;CODE FOR UPWARDS MAP IN MEM MGMT PDR'S
                        000006          RW = 6                  ;CODE FOR READ/WRITE IN MEM MGMT PDR'S

                                        ;*  PARITY MEMORY DEFINITIONS.
                        000001          AE=1                            ;PARITY ACTION ENABLE
                        000114          PARVEC=114                      ;PARITY TRAP VECTOR

                                        ;*  MISCELLANEOUS ASSIGNMENTS
                        017777          MASK4K= 17777                   ;MASK FOR 4K ADDRESS BANK BOUNDARY.
                        037777          MASK8K= 37777                   ;MASK FOR 8K ADDRESS BANK BOUNDARY
                        000007          MFPT=   7                       ;MOVE FROM PROCESSOR TYPE TO R0


                                        .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
                                        .SBTTL  STARTING ADDRESS(ES)
000200  000137  003742                  JMP     @#START ;;JUMP TO STARTING ADDRESS OF PROGRAM
000204  000167  000070                  JMP     RESTAR          ;RESTART ADDRESS, USING PREVIOUS PARAMETERS.

                        000004          .=ERRVEC
000004  023252          .WORD   ERRTRP
000006  000000          .WORD   0

                                        .SBTTL  ACT11 HOOKS

                                        ;;**************************************************************
                                        ;HOOKS REQUIRED BY ACT11
                        000010          $SVPC=.                 ;SAVE PC
                        000046          .=46
000046  012560          $ENDAD                  ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
                        000052          .=52
000052  040000          .WORD   BIT14           ;;2)SET LOC.52 TO BIT14
                        000010          .=$SVPC                 ;; RESTORE PC
```

```
                000100                  .=100
000100  000104  000340  000002  .WORD   104,340,2                ;IF BEVENT IS ENABLED
                                                                ;IGNORE ITS INTERRUPT,DO RTI

                000300                  .=300
                                ;;*****************************************************************
                                ;*  THE FOLLOWING ROUTINES ARE LOCATED IN THE VECTOR AREA (0-1000) SO THAT
                                ;*   THEY CAN BE PROTECTED BY SELECTING SW05 (SEE DOCUMENT FOR USE OF SW05).
                                ;;*****************************************************************
000300  005005                  RESTAR: CLR     R5              ;CLEAR FLAG TO INDICATE RESTART.
000302  000401                          BR      REST1           ;GO RESTORE PROGRAM BEFORE RESTARTING.
000304  010705                  RESTOR: MOV     PC,     R5      ;PUT DATA INTO FLAG FOR RESTORE.
000306  012706  001070          REST1:  MOV     #$CMTAG,SP      ;SET UP THE STACK POINTER.
000312  005767  000216                  TST     PRGMAP          ;CHECK IF THE MEMORY HAS BEEN MAPPED.
000316  001002                          BNE     REST2           ;BR IF MEMORY MAPPED.
000320  000167  003432                  JMP     STARTA          ;GO START,IF NOT MAPPED
000324  005767  000206          REST2:  TST     MMAVA           ;CHECK IF MEM MGMT AVAILABLE.
000330  001454                          BEQ     10$             ;BR IF NO MEM MGMT.
000332  032737  000001  177572          BIT     #BIT0,  @#SR0   ;CHECK IF MEM MGMT ACTIVE.
000340  001027                          BNE     2$              ;BR IF MEM MGMT ALREADY SET UP.
000342  012700  172300                  MOV     #KIPDR0,R0      ;POINT TO FIRST MEM MGMT DDATA REC.
000346  012701  000010                  MOV     #8.,    R1      ;SET UP COUNTER.
000352  012720  077406          1$:     MOV     #077406,(R0)+   ;MAP FIRST 28K 1-FOR-1.
000356  005301                          DEC     R1              ;COUNT REGISTERS.
000360  001374                          BNE     1$              ;BR IF MORE REG.
000362  012700  172340                  MOV     #KIPAR0,R0      ;POINT TO FIRST MEM MGMT ADDRESS REG.
000366  005020                          CLR     (R0)+           ;PAR0 MAPPED INTO BANK0.
000370  012720  000200                  MOV     #200,   (R0)+   ;PAR1 MAPPED INTO BANK1.
000374  005020                          CLR     (R0)+           ;PAR2 CLEARED
000376  005020                          CLR     (R0)+           ;PAR3 CLEARED
000400  005020                          CLR     (R0)+           ;PAR4 CLEARED
000402  005020                          CLR     (R0)+           ;PAR5 CLEARED
000404  005020                          CLR     (R0)+           ;PAR6 CLEARED
000406  012720  177600                  MOV     #177600,(R0)+   ;PAR7 MAPPED INTO LAST BANK.
000412  012737  000001  177572          MOV     #BIT0,  @#SR0   ;ENABLE MEM MGMT.
000420  005000                  2$:     CLR     R0              ;INIT TEMP PAR REG.
000422  016701  000106                  MOV     PRGMAP, R1      ;GET THE PROGRAM MAP...LO 64K.
000426  006201                  3$:     ASR     R1              ;SHIFT THE MAP POINTER...
000430  103403                          BCS     4$              ;BR WHEN FIRST 8K BANK FOUND.
000432  062700  000400                  ADD     #400,   R0      ;UPDATE TMP PAR TO NEXT 8K BANK.
000436  000773                          BR      3$              ;NEXT 8K BANK
000440  010037  172340          4$:     MOV     R0,     @#KIPAR0 ;PUT TEMP PAR INTO FIRST PAR.
000444  000137  000450                  JMP     @#5$            ;JUMP INTO PROGRAM IF NOT THERE ALREADY.
000450  062700  000200          5$:     ADD     #200,   R0      ;KEEP UPDATING TEMP PAR REG.
000454  010037  172342                  MOV     R0,     @#KIPAR1 ;SET UP SECOND PROGRAM BANK POINTER.
000460  000410                          BR      20$             ;BR TO RELOCATE SECTION.
000462  016700  000044          10$:    MOV     RELOCF, R0      ;GET RELOCATION FACTOR.
000466  062700  001070                  ADD     #$CMTAG,R0      ;SET UP STACK POINTER.
000472  010006                          MOV     R0,     SP      ;SET STACK TO RELOCATE PROGRAM.
000474  062700  177412                  ADD     #20$-#$CMTAG,R0 ;ADJUST R0 TO RELOCATED "20$" ADDRESS.
000500  000110                          JMP     (R0)            ;GO TO "20$" (RELOCATED).
000502  022767  000001  000024  20$:    CMP     #1,     PRGMAP  ;CHECK IF PROGRAM IS IN 8K BANK 0 .
000510  001402                          BEQ     21$             ;BR IF IN 8K BANK 0.
000512  004767  014370                  JSR     PC,     RELO    ;RELOCATE THE PROGRAM BACK TO BANK 0.
000516  005067  000436          21$:    CLR     $TIMES          ;CLEAN UP BEFORE STARTING.
000522  105067  000344                  CLRB    $TSTNM
```

```
  271  000526  000167  006130                    JMP     START1          ;RESTART WITH PREVIOUSLY SELECTED PARAMETERS.
  272
  273                                           ;* THE FOLLOWING LOCATIONS ARE USED BY THE ABOVE ROUTINE AND MUST BE LOCATED
  274                                           ;*  BELOW 1000 TO INSURE CORRECT OPERATION UNDER THE WIDEST VARIETY OF
  275                                           ;*  CIRCUMSTANCES.
  276  000532  000000                          RELOCF: .WORD   0       ;CONTAINS RELOCATION FACTOR (NO MEM MGMT)
  277  000534  000000                          PRGMAP: .WORD   0       ;PROGRAM MAP - WHERE THE PROGRAM IS LOCATED
  278  000536  000000                          MMAVA:  .WORD   0       ;MEMORY MANAGEMENT AVAILABLE FLAG.
  279                                                                  ;BIT  0 - 1     MMU AVAILABLE
  280                                                                  ;BIT 15 - 1     22 BIT ADDRESSING AVAILABLE
```

```
                        .SBTTL  POWER DOWN AND UP ROUTINES

                        ;;*****************************************************************
                        ;POWER DOWN ROUTINE
000540  012737  000706  000024  $PWRDN: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
000546  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
000554  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
000556  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
000560  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
000562  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
000564  010446                          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
000566  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
000570  017746  000334                  MOV     @SWR,-(SP)      ;;PUSH @SWR ON STACK
000574  010667  000112                  MOV     SP,$SAVR6       ;;SAVE SP
000600  012737  000612  000024          MOV     #$PWRUP,@#PWRVEC ;;SET UP VECTOR
000606  000000                          HALT
000610  000776                          BR      .-2             ;;HANG UP

                        ;;*****************************************************************
                        ;POWER UP ROUTINE
000612  012737  000706  000024  $PWRUP: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
000620  016706  000066                  MOV     $SAVR6,SP       ;;GET SP
000624  005067  000062                  CLR     $SAVR6          ;;WAIT LOOP FOR THE TTY
000630  005267  000056          1$:     INC     $SAVR6          ;;WAIT FOR THE INC
000634  001375                          BNE     1$              ;;OF  WORD
000636  012677  000266                  MOV     (SP)+,@SWR      ;;POP STACK INTO @SWR
000642  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
000644  012604                          MOV     (SP)+,R4        ;;POP STACK INTO R4
000646  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
000650  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
000652  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
000654  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
000656  012737  000540  000024          MOV     #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
000664  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
000672  004567  020676                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
000676  023761                  $PWRMG: .WORD   PWRMSG          ;;POWER FAIL MESSAGE POINTER
000700  012716                          MOV     (PC)+,(SP)      ;;RESTART AT RESTART
000702  000300                  $PWRAD: .WORD   RESTART         ;;RESTART ADDRESS
000704  000002                          RTI
000706  000000                  $ILLUP: HALT                    ;;THE POWER UP SEQUENCE WAS STARTED
000710  000776                          BR      .-2             ;; BEFORE THE POWER DOWN WAS COMPLETE
000712  000000                  $SAVR6: 0                       ;;PUT THE SP HERE
```

```
                                .SBTTL  COMMON TAGS

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

            001070                      .=1070
001070                          $CMTAG:                         ;;START OF COMMON TAGS
001070  000000                          .WORD   0
001072     000                  $TSTNM: .BYTE   0               ;;CONTAINS THE TEST NUMBER
001073     000                  $ERFLG: .BYTE   0               ;;CONTAINS ERROR FLAG
001074  000000                  $ICNT:  .WORD   0               ;;CONTAINS SUBTEST ITERATION COUNT
001076  000000                  $LPADR: .WORD   0               ;;CONTAINS SCOPE LOOP ADDRESS
001100  000000                  $LPERR: .WORD   0               ;;CONTAINS SCOPE RETURN FOR ERRORS
001102  000000                  $ERTTL: .WORD   0               ;;CONTAINS TOTAL ERRORS DETECTED
001104     000                  $ITEMB: .BYTE   0               ;;CONTAINS ITEM CONTROL BYTE
001105     001                  $ERMAX: .BYTE   1               ;;CONTAINS MAX. ERRORS PER TEST
001106  000000                  $ERRPC: .WORD   0               ;;CONTAINS PC OF LAST ERROR INSTRUCTION
001110  000000                  $GDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'GOOD' DATA
001112  000000                  $BDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'BAD' DATA
001114  000000                  $GDDAT: .WORD   0               ;;CONTAINS 'GOOD' DATA
001116  000000                  $BDDAT: .WORD   0               ;;CONTAINS 'BAD' DATA
001120  000000                          .WORD   0               ;;RESERVED--NOT TO BE USED
001122  000000                          .WORD   0
001124     000                  $AUTOB: .BYTE   0               ;;AUTOMATIC MODE INDICATOR
001125     000                  $INTAG: .BYTE   0               ;;INTERRUPT MODE INDICATOR
001126  000000                          .WORD   0
001130  177570                  SWR:    .WORD   DSWR            ;;ADDRESS OF SWITCH REGISTER
001132  177570                  DISPLAY: .WORD  DDISP           ;;ADDRESS OF DISPLAY REGISTER
001134  177560                  $TKS:   177560                  ;;TTY KBD STATUS
001136  177562                  $TKB:   177562                  ;;TTY KBD BUFFER
001140  177564                  $TPS:   177564                  ;;TTY PRINTER STATUS REG. ADDRESS
001142  177566                  $TPB:   177566                  ;;TTY PRINTER BUFFER REG. ADDRESS
001144     000                  $NULL:  .BYTE   0               ;;CONTAINS NULL CHARACTER FOR FILLS
001145     002                  $FILLS: .BYTE   2               ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
001146     012                  $FILLC: .BYTE   12              ;;INSERT FILL CHARS. AFTER A 'LINE FEED''
001147     000                  $TPFLG: .BYTE   0               ;;''TERMINAL AVAILABLE'' FLAG (BIT<07>=0=YES)
001150  000000                  $TMP0:  .WORD   0               ;;USER DEFINED
001152  000000                  $TMP1:  .WORD   0               ;;USER DEFINED
001154  000000                  $TMP2:  .WORD   0               ;;USER DEFINED
001156  000000                  $TMP3:  .WORD   0               ;;USER DEFINED
001160  000000                  $TIMES: 0                       ;;MAX. NUMBER OF ITERATIONS
001162  000000                  $ESCAPE:0                       ;;ESCAPE ON ERROR ADDRESS
001164  177607  000377          $BELL:  .ASCIZ  <207><377><377> ;;CODE FOR BELL
001170     077                  $QUES:  .ASCII  /?/             ;;QUESTION MARK
001171     015                  $CRLF:  .ASCII  <15>            ;;CARRIAGE RETURN
001172  000012                  $LF:    .ASCIZ  <12>            ;;LINE FEED
                                ;;*****************************************************************
                                .SBTTL  APT MAILBOX-ETABLE

                                ;;*****************************************************************
                                .EVEN
001174                          $MAIL:                  ;;APT MAILBOX
001174  000000                  $MSGTY: .WORD   AMSGTY  ;;MESSAGE TYPE CODE
001176  000000                  $FATAL: .WORD   AFATAL  ;;FATAL ERROR NUMBER
001200  000000                  $TESTN: .WORD   ATESTN  ;;TEST NUMBER
```

```
  379  001202  000000                $PASS:  .WORD   APASS   ;;PASS COUNT
  380  001204  000000                $DEVCT: .WORD   ADEVCT  ;;DEVICE COUNT
  381  001206  000000                $UNIT:  .WORD   AUNIT   ;;I/O UNIT NUMBER
  382  001210  000000                $MSGAD: .WORD   AMSGAD  ;;MESSAGE ADDRESS
  383  001212  000000                $MSGLG: .WORD   AMSGLG  ;;MESSAGE LENGTH
  384  001214                         $ETABLE:                ;;APT ENVIRONMENT TABLE
  385  001214      000                $ENV:   .BYTE   AENV    ;;ENVIRONMENT BYTE
  386  001215      000                $ENVM:  .BYTE   AENVM   ;;ENVIRONMENT MODE BITS
  387  001216  000000                $SWREG: .WORD   ASWREG  ;;APT SWITCH REGISTER
  388  001220  000000                $USWR:  .WORD   AUSWR   ;;USER SWITCHES
  389  001222  000000                $CPUOP: .WORD   ACPUOP  ;;CPU TYPE,OPTIONS
  390                                 ;*                     BITS 15-11=CPU TYPE
  391                                 ;*                             11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
  392                                 ;*                             11/70=06,PDQ=07,Q=10
  393                                 ;*                     BIT 10=REAL TIME CLOCK
  394                                 ;*                     BIT  9=FLOATING POINT PROCESSOR
  395                                 ;*                     BIT  8=MEMORY MANAGEMENT
  396  001224      000                $MAMS1: .BYTE   AMAMS1  ;;HIGH ADDRESS,M.S. BYTE
  397  001225      000                $MTYP1: .BYTE   AMTYP1  ;;MEM. TYPE,BLK#1
  398                                 ;*                     MEM.TYPE BYTE   --  (HIGH BYTE)
  399                                 ;*                             900 NSEC CORE=001
  400                                 ;*                             300 NSEC BIPOLAR=002
  401                                 ;*                             500 NSEC MOS=003
  402  001226  000000                $MADR1: .WORD   AMADR1  ;;HIGH ADDRESS,BLK#1
  403                                 ;*                       MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF ''TYPE'' ABOVE
  404  001230      000                $MAMS2: .BYTE   AMAMS2  ;;HIGH ADDRESS,M.S. BYTE
  405  001231      000                $MTYP2: .BYTE   AMTYP2  ;;MEM.TYPE,BLK#2
  406  001232  000000                $MADR2: .WORD   AMADR2  ;;MEM.LAST ADDRESS,BLK#2
  407  001234      000                $MAMS3: .BYTE   AMAMS3  ;;HIGH ADDRESS,M.S.BYTE
  408  001235      000                $MTYP3: .BYTE   AMTYP3  ;;MEM.TYPE,BLK#3
  409  001236  000000                $MADR3: .WORD   AMADR3  ;;MEM.LAST ADDRESS,BLK#3
  410  001240      000                $MAMS4: .BYTE   AMAMS4  ;;HIGH ADDRESS,M.S.BYTE
  411  001241      000                $MTYP4: .BYTE   AMTYP4  ;;MEM.TYPE,BLK#4
  412  001242  000000                $MADR4: .WORD   AMADR4  ;;MEM.LAST ADDRESS,BLK#4
  413  001244  000000                $VECT1: .WORD   AVECT1  ;;INTERRUPT VECTOR#1,BUS PRIORITY#1
  414  001246  000000                $VECT2: .WORD   AVECT2  ;;INTERRUPT VECTOR#2BUS PRIORITY#2
  415  001250  000000                $BASE:  .WORD   ABASE   ;;BASE ADDRESS OF EQUIPMENT UNDER TEST
  416  001252  000000                $DEVM:  .WORD   ADEVM   ;;DEVICE MAP
  417  001254  000000                $CDW1:  .WORD   ACDW1   ;;CONTROLLER DESCRIPTION WORD#1
  418  001256  000000                $CDW2:  .WORD   ACDW2   ;;CONTROLLER DESCRIPTION WORD#2
  419  001260  000000                $DDW0:  .WORD   ADDW0   ;;DEVICE DESCRIPTOR WORD#0
  420  001262  000000                $DDW1:  .WORD   ADDW1   ;;DEVICE DESCRIPTOR WORD#1
  421  001264  000000                $DDW2:  .WORD   ADDW2   ;;DEVICE DESCRIPTOR WORD#2
  422  001266  000000                $DDW3:  .WORD   ADDW3   ;;DEVICE DESCRIPTOR WORD#3
  423  001270  000000                $DDW4:  .WORD   ADDW4   ;;DEVICE DESCRIPTOR WORD#4
  424  001272  000000                $DDW5:  .WORD   ADDW5   ;;DEVICE DESCRIPTOR WORD#5
  425  001274  000000                $DDW6:  .WORD   ADDW6   ;;DEVICE DESCRIPTOR WORD#6
  426  001276  000000                $DDW7:  .WORD   ADDW7   ;;DEVICE DESCRIPTOR WORD#7
  427  001300  000000                $DDW8:  .WORD   ADDW8   ;;DEVICE DESCRIPTOR WORD#8
  428  001302  000000                $DDW9:  .WORD   ADDW9   ;;DEVICE DESCRIPTOR WORD#9
  429  001304  000000                $DDW10: .WORD   ADDW10  ;;DEVICE DESCRIPTOR WORD#10
  430  001306  000000                $DDW11: .WORD   ADDW11  ;;DEVICE DESCRIPTOR WORD#11
  431  001310  000000                $DDW12: .WORD   ADDW12  ;;DEVICE DESCRIPTOR WORD#12
  432  001312  000000                $DDW13: .WORD   ADDW13  ;;DEVICE DESCRIPTOR WORD#13
  433  001314  000000                $DDW14: .WORD   ADDW14  ;;DEVICE DESCRIPTOR WORD#14
  434  001316  000000                $DDW15: .WORD   ADDW15  ;;DEVICE DESCRIPTOR WORD#15
```

```
                                        $ETEND:
                                        .MEXIT
                                        .SBTTL  APT PARAMETER BLOCK

                                        ;;*****************************************************************
                                        ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
                                        ;;************************************* **************************
                001320                          .$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          001320                          $APTHDR ;;POINT TO APT HEADER BLOCK
                001320                          .=.$X   ;;RESET LOCATION COUNTER
                                        ;;*****************************************************************
                                        ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
                                        ;INTERFACE SPEC.

001320                                  $APTHD:
001320  000000                          $HIBTS: .WORD   0       ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
001322  001174                          $MBADR: .WORD   $MAIL   ;;ADDRESS OF APT MAILBOX (BITS 0-15)
001324  001604                          $TSTM:  .WORD   900.    ;;RUN TIM OF LONGEST TEST
001326  002260                          $PASTM: .WORD   1200.   ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
001330  000000                          $UNITM: .WORD   0.      ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
001332  000052                                  .WORD   $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
                                        .SBTTL  APT STATISTICS TABLE

                                        ;;*****************************************************************
001334                                  $ASTAT:
001334  177777  000000                          .WORD   -1,0
001340  177777  000000                          .WORD   -1,0
001344  177777  000000                          .WORD   -1,0
001350  177777  000000                          .WORD   -1,0
001354  177777  000000                          .WORD   -1,0
001360  177777  000000                          .WORD   -1,0
001364  177777  000000                          .WORD   -1,0
001370  177777  000000                          .WORD   -1,0
001374  177777  000000                          .WORD   -1,0
001400  177777  000000                          .WORD   -1,0
001404  177777  000000                          .WORD   -1,0
001410  177777  000000                          .WORD   -1,0
001414  177777  000000                          .WORD   -1,0
001420  177777  000000                          .WORD   -1,0
001424  177777  000000                          .WORD   -1,0
001430  177777  000000                          .WORD   -1,0
001434  177777  000000                          .WORD   -1,0
001440  177777  000000                          .WORD   -1,0
001444  177777  000000                          .WORD   -1,0
001450  177777  000000                          .WORD   -1,0
001454  177777  000000                          .WORD   -1,0
001460  177777  000000                          .WORD   -1,0
001464  177777  000000                          .WORD   -1,0
001470  177777  000000                          .WORD   -1,0
001474  177777  000000                          .WORD   -1,0
001500  177777                          $ASTEND:        -1
001502  001334                          $APTR:  $ASTAT
```

```
                                ;;*****************************************************************
                                ;*THE FOLLOWING TAGS ARE USER DEFINED
                                ;;*****************************************************************
001504  000000                  $VERPC: .WORD   0       ;VIRTUAL PC LOCATION FOR ERROR TYPEOUT ROUTINE ($ERTYP).
001506  070032                  RESRVD: .WORD   070032  ;CORE PARITY REG BITS RESERVED FOR FUTURE USE.
                                                        ;NOTE:  FOR MS11 MEMORY WITH PARITY, CHANGE TO 077772.
001510  000000                  LMAD:   .WORD   0       ;LAST CONTIGUOUS MEMORY ADDRESS (+2)
001512  000000                  LDDISP: .WORD   0       ;CONTAINS DISPLAY REGISTER IMAGE

001514                          MEMMAP:                 ;MEMORY MAP - EACH BIT CORRESPONDS TO 8K

001514  000000                          .WORD   0       ;1ST  WORD MAP...128KW
001516  000000                          .WORD   0       ;2ND  WORD MAP...256KW
001520  000000                          .WORD   0       ;3RD  WORD MAP...384KW
001522  000000                          .WORD   0       ;4TH  WORD MAP...512KW
001524  000000                          .WORD   0       ;5TH  WORD MAP...640KW
001526  000000                          .WORD   0       ;6TH  WORD MAP...768KW
001530  000000                          .WORD   0       ;7TH  WORD MAP...896KW
001532  000000                          .WORD   0       ;8TH  WORD MAP..1024KW
001534  000000                          .WORD   0       ;9TH  WORD MAP..1152KW
001536  000000                          .WORD   0       ;10TH WORD MAP..1280KW
001540  000000                          .WORD   0       ;11TH WORD MAP..1408KW
001542  000000                          .WORD   0       ;12TH WORD MAP..1536KW
001544  000000                          .WORD   0       ;13TH WORD MAP..1664KW
001546  000000                          .WORD   0       ;14TH WORD MAP..1792KW
001550  000000                          .WORD   0       ;15TH WORD MAP..1920KW
001552  000000                          .WORD   0       ;16TH WORD MAP..2048KW

001554                          TSTMAP:                 ;TEST MAP - WHICH BANKS ARE SELECTED FOR TEST.

001554  000000                          .WORD   0       ;1ST  WORD MAP...128KW
001556  000000                          .WORD   0       ;2ND  WORD MAP...256KW
001560  000000                          .WORD   0       ;3RD  WORD MAP...384KW
001562  000000                          .WORD   0       ;4TH  WORD MAP...512KW
001564  000000                          .WORD   0       ;5TH  WORD MAP...640KW
001566  000000                          .WORD   0       ;6TH  WORD MAP...768KW
001570  000000                          .WORD   0       ;7TH  WORD MAP...896KW
001572  000000                          .WORD   0       ;8TH  WORD MAP..1024KW
001574  000000                          .WORD   0       ;9TH  WORD MAP..1152KW
001576  000000                          .WORD   0       ;10TH WORD MAP..1280KW
001600  000000                          .WORD   0       ;11TH WORD MAP..1408KW
001602  000000                          .WORD   0       ;12TH WORD MAP..1536KW
001604  000000                          .WORD   0       ;13TH WORD MAP..1664KW
001606  000000                          .WORD   0       ;14TH WORD MAP..1792KW
001610  000000                          .WORD   0       ;15TH WORD MAP..1920KW
001612  000000                          .WORD   0       ;16TH WORD MAP..2048KW

001614                          SAVTST:                 ;SAVED TEST MAP - USED DURING FIRST PASS TO ONLY
                                                        ; TEST EACH BANK ONCE.

001614  000000                          .WORD   0       ;1ST  WORD MAP...128KW
001616  000000                          .WORD   0       ;2ND  WORD MAP...256KW
001620  000000                          .WORD   0       ;3RD  WORD MAP...384KW
001622  000000                          .WORD   0       ;4TH  WORD MAP...512KW
001624  000000                          .WORD   0       ;5TH  WORD MAP...640KW
001626  000000                          .WORD   0       ;6TH  WORD MAP...768KW
```

```
                                        .WORD   0       ;7TH  WORD MAP...896KW
                                        .WORD   0       ;8TH  WORD MAP..1024KW
                                        .WORD   0       ;9TH  WORD MAP..1152KW
                                        .WORD   0       ;10TH WORD MAP..1280KW
                                        .WORD   0       ;11TH WORD MAP..1408KW
                                        .WORD   0       ;12TH WORD MAP..1536KW
                                        .WORD   0       ;13TH WORD MAP..1664KW
                                        .WORD   0       ;14TH WORD MAP..1792KW
                                        .WORD   0       ;15TH WORD MAP..1920KW
                                        .WORD   0       ;16TH WORD MAP..2048KW

                        PMEMAP:                         ;PARITY MAP - WHICH BANKS HAVE MEMORY PARITY

                                        .WORD   0       ;1ST  WORD MAP...128KW
                                        .WORD   0       ;2ND  WORD MAP...256KW
                                        .WORD   0       ;3RD  WORD MAP...384KW
                                        .WORD   0       ;4TH  WORD MAP...512KW
                                        .WORD   0       ;5TH  WORD MAP...640KW
                                        .WORD   0       ;6TH  WORD MAP...768KW
                                        .WORD   0       ;7TH  WORD MAP...896KW
                                        .WORD   0       ;8TH  WORD MAP..1024KW
                                        .WORD   0       ;9TH  WORD MAP..1152KW
                                        .WORD   0       ;10TH WORD MAP..1280KW
                                        .WORD   0       ;11TH WORD MAP..1408KW
                                        .WORD   0       ;12TH WORD MAP..1536KW
                                        .WORD   0       ;13TH WORD MAP..1664KW
                                        .WORD   0       ;14TH WORD MAP..1792KW
                                        .WORD   0       ;15TH WORD MAP..1920KW
                                        .WORD   0       ;16TH WORD MAP..2048KW

                        BITPT:                          ;POINTER TO CURRENT 8K BANK OF MEMORY

                                        .WORD   0       ;1ST  WORD MAP...128KW
                                        .WORD   0       ;2ND  WORD MAP...256KW
                                        .WORD   0       ;3RD  WORD MAP...384KW
                                        .WORD   0       ;4TH  WORD MAP...512KW
                                        .WORD   0       ;5TH  WORD MAP...640KW
                                        .WORD   0       ;6TH  WORD MAP...768KW
                                        .WORD   0       ;7TH  WORD MAP...896KW
                                        .WORD   0       ;8TH  WORD MAP..1024KW
                                        .WORD   0       ;9TH  WORD MAP..1152KW
                                        .WORD   0       ;10TH WORD MAP..1280KW
                                        .WORD   0       ;11TH WORD MAP..1408KW
                                        .WORD   0       ;12TH WORD MAP..1536KW
                                        .WORD   0       ;13TH WORD MAP..1664KW
                                        .WORD   0       ;14TH WORD MAP..1792KW
                                        .WORD   0       ;15TH WORD MAP..1920KW
                                        .WORD   0       ;16TH WORD MAP..2048KW


                        MMORE:  .WORD   0       ;LOOP ADDRESS FOR MULTIPLE BLOCK TESTING.
                                                ;SET UP BY ''INITMM'' AND ''INITDN'' ROUTINES.
                                                ;USED BY 'MMUP'' AND 'MMDOWN'' ROUTINES.
                        SELFLG: .BYTE   0       ;OPERATOR SELECTED PARAMETERS FLAG.  (SA=204)
                        FLAG8K: .BYTE   0       ;8K BLOCK INDICATOR.  USED IN ''INITMM'' AND 'MMUP''.
```

```
                                OEFLG:  .BYTE   0       ;ODD/EVEN FLAG USED IN PARITY MEMORY BYTE TEST.
                                        .EVEN
                                FSTADR: .WORD   0       ;FIRST VIRTUAL ADDRESS TO BE TESTED.
                                                        ;FIRST ADDRESS IS USER SELECTABLE.
                                                        ; TO BREAK TO 'MMUP'' TO FIND LAST ADDRESS.
                                .CONST: .WORD   0       ;USER SELECTABLE CONSTANT DATA.
                                WWP:    .WORD   4       ;WRITE WRONG PARITY COMMAND
                                TEMP:   .WORD   0       ;TEMPORARY STORAGE
                                TEMP1:  .WORD   0       ;TEMPORARY STORAGE
                                FLG30K: .WORD   0       ;30K MEMORY FLAG
                                LSIFLG: .WORD   0       ;LSI-11 PROCESSOR FLAG
                                PLUS1:  .WORD   0       ;DATA BIT USED FOR FIRST ENTRY INTO
                                                        ;MMUP AND MMDOWN ROUTINES.


                                ;;*************************************************************
                                ;*  RELATIVE ADDRESSING TABLE.
                                ;*  THE FOLLOWING LOCATIONS ARE MODIFIED AT RELOCATION TIME TO ALLOW
                                ;*  RELATIVE ADDRESSING TO GET THE RELOCATED VALUE OF THE ARGUMENT TAGS.
                                ;;*********************************************************  ******
                                RADTAB:
                                .STACK: $CMTAG          ;STACK POINTER INITIAL ADDRESS.
                                .MEMMAP:        MEMMAP  ;AUTO. MEMORY SIZING MAP
                                .SAVTST:        SAVTST  ;SIZED MEMORY MAP FROM AUTO. OR USER
                                .TSTMAP:        TSTMAP  ;MEMORY MAP OF 8K BANKS TO TEST
                                .PMEMAP:        PMEMAP  ;PARITY MAP OF 8K BANKS
                                .BITPT: BITPT           ;BIT POINTER FOR BLOCK UNDER TEST
                                .RESRV: RESRVD          ;PARITY REGISTER RESERVED BIT MASK ADDRESS.
                                .MPR0:  MPR0            ;MEMORY PARITY REGISTER TABLE ADDRESS.
                                .MPRX:  MPRX            ;MEMORY PARITY REGISTER EXIST TABLE ADDRESS.
                                .PBTRP: PBTRP           ;PARITY BYTE TEST TRAP ROUTINE ADDRESS.
                                .MPPAT: MPPATS          ;MEMORY PARITY PATTERN TABLE ADDRESS.
                                .PESRV: PESRV           ;MEMORY PARITY ERROR TRAP ROUTINE ADDRESS.
                                .ERRTB: $ERRTB          ;ERROR TYPEOUT TABLE PONTER.
                                .EIGHT: 8.              ;DECIMAL TYPE ROUTINE COUNT DESIGNATOR.
                                .TST32: TST32           ;SCOPE ABORT ADR FOR WHEN NO MEM AVA FOR TEST.
                                ;;*************************************************************
                                ;*  DATA CONTAINERS FOR ERROR PRINTOUT.
                                ;;*************************************************************
                                DT1:    $ERRPC,$GDADR,$GDDAT,$BDDAT,0

                                DT2:    $VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT,0

                                DT12:   $VERPC,$ERRPC,$GDADR,$GDDAT,0

                                DT14:   $VERPC,$ERRPC,$TMP0,$GDADR,0

                                DT15:   $VERPC,$ERRPC,$GDADR,$TMP0,$GDDAT,$BDDAT,0

                                DT21:   $VERPC,$ERRPC,$TMP0,$GDADR,$GDDAT,$BDDAT,0

                                DT23:   $VERPC,$ERRPC,$GDADR,$BDADR,$GDDAT,$BDDAT,0

                                DT24:   $VERPC,$ERRPC,$BDADR,0
```

```
 659  002172  000000
 660  002174  001504  001106  001112  DT25:   $VERPC,$ERRPC,$BDADR,$TMP0,$TMP1,0
 661  002202  001150  001152  000000
 662  002210  001504  001106  001150  DT26:   $VERPC,$ERRPC,$TMP0,$TMP1,0
 663  002216  001152  000000
 664  002222  001150  001152  001110  DT30:   $TMP0,$TMP1,$GDADR,$BDDAT,0
 665  002230  001116  000000
 666  002234  001156  000000          DT31:   $TMP3,0
 667  002240  177777                          .WORD   -1      ;TABLE TERMINATOR.
 668
 669                                  .SBTTL  MEMORY PARITY PATTERNS TABLE
 670                                  ;;*****************************************************************
 671                                  ;THE FOLLOWING ARE THE PARITY PATTERNS EXERCISED THRUOUT MEMORY
 672                                  ;;*****************************************************************
 673
 674  002242  125325                  MPPATS: 125325                  ;EVEN,ODD
 675  002244  152652                          152652                  ;ODD,EVEN
 676  002246  052452                          052452                  ;EVEN,ODD
 677  002250  025125                          025125                  ;ODD,EVEN
 678  002252  102070                          102070                  ;EVEN,EVEN
 679  002254  072527                          072527                  ;ODD,ODD
 680  002256  177777                          177777                  ;EVEN,EVEN
 681  002260  107030                          107030                  ;ODD,ODD
 682  002262  152525                          152525                  ;ODD,EVEN
 683  002264  000000                          0                       ;EXTRA PATTERN HOLDER FOR
 684                                                                  ;FUTURE USE
 685  002266  000000                  MPEND:  0                       ;TABLE TERMINATOR
 686
 687                                  .SBTTL  MEMORY PARITY REGISTER ADDRESS TABLE
 688                                  ;;//////////////////////////////////////////////////////////////////
 689                                  ;*  THE FOLLOWING REPRESENTS THE MEMORY PARITY REGISTER ADDRESS TABLE
 690                                  ;*  FROM WHICH PARITY MEMORY IS ADDRESSED & CONTROLLED:
 691                                  ;*
 692                                  ;*  THE LEAST SIGNIFICANT BIT IN THE DEVICE ADDRESS IS SET TO A ONE (1)
 693                                  ;*  IF THE CONTROL IS FOUND NOT TO BE PRESENT.  THE MEMORY PRESENT UNDER
 694                                  ;*  THE CONTROL OF EACH CONTROLLER IS REPRESENTED BY 16 WORDS FOLLOWING
 695                                  ;*  THE DEVICE ADDRESS.  FOR NON-22BIT MEMORY ONLY 1 WORD IS USED
 696                                  ;*  (EACH BIT REPRESENTING A 8K BLOCK,  I.E.  FIRST WORD BIT0 = 0 - 8K,
 697                                  ;*      BIT  3 =        24-28KW         16 BIT ADDRESSING
 698                                  ;*      BIT 15 =        120-124KW       18 BIT ADDRESSING
 699                                  ;;//////////////////////////////////////////////////////////////////
 700  002270  172101                  MPR0:   172100 +1               ;PARITY STATUS REGISTER
 701  002272  000000                          .WORD   0               ;1ST  WORD CONTROL MAP...128KW
 702  002274  000000                          .WORD   0               ;2ND  WORD CONTROL MAP...256KW
 703  002276  000000                          .WORD   0               ;3RD  WORD CONTROL MAP...384KW
 704  002300  000000                          .WORD   0               ;4TH  WORD CONTROL MAP...512KW
 705  002302  000000                          .WORD   0               ;5TH  WORD CONTROL MAP...640KW
 706  002304  000000                          .WORD   0               ;6TH  WORD CONTROL MAP...768KW
 707  002306  000000                          .WORD   0               ;7TH  WORD CONTROL MAP...896KW
 708  002310  000000                          .WORD   0               ;8TH  WORD CONTROL MAP..1024KW
 709  002312  000000                          .WORD   0               ;9TH  WORD CONTROL MAP..1152KW
 710  002314  000000                          .WORD   0               ;10TH WORD CONTROL MAP..1280KW
 711  002316  000000                          .WORD   0               ;11TH WORD CONTROL MAP..1408KW
 712  002320  000000                          .WORD   0               ;12TH WORD CONTROL MAP..1536KW
 713  002322  000000                          .WORD   0               ;13TH WORD CONTROL MAP..1664KW
 714  002324  000000                          .WORD   0               ;14TH WORD CONTROL MAP..1792KW
```

```
715  002326  000000                    .WORD   0       ;15TH WORD CONTROL MAP..1920KW
716  002330  000000                    .WORD   0       ;16TH WORD CONTROL MAP..2048KW
717  002332  000000                    .WORD   0       ;MASK FOR UNUSED PARITY CSR BITS
718  002334  172103            MPR1:   172102 +1       ;PARITY STATUS REGISTER
719  002336  000000                    .WORD   0       ;1ST  WORD CONTROL MAP...128KW
720  002340  000000                    .WORD   0       ;2ND  WORD CONTROL MAP...256KW
721  002342  000000                    .WORD   0       ;3RD  WORD CONTROL MAP...384KW
722  002344  000000                    .WORD   0       ;4TH  WORD CONTROL MAP...512KW
723  002346  000000                    .WORD   0       ;5TH  WORD CONTROL MAP...640KW
724  002350  000000                    .WORD   0       ;6TH  WORD CONTROL MAP...768KW
725  002352  000000                    .WORD   0       ;7TH  WORD CONTROL MAP...896KW
726  002354  000000                    .WORD   0       ;8TH  WORD CONTROL MAP..1024KW
727  002356  000000                    .WORD   0       ;9TH  WORD CONTROL MAP..1152KW
728  002360  000000                    .WORD   0       ;10TH WORD CONTROL MAP..1280KW
729  002362  000000                    .WORD   0       ;11TH WORD CONTROL MAP..1408KW
730  002364  000000                    .WORD   0       ;12TH WORD CONTROL MAP..1536KW
731  002366  000000                    .WORD   0       ;13TH WORD CONTROL MAP..1664KW
732  002370  000000                    .WORD   0       ;14TH WORD CONTROL MAP..1792KW
733  002372  000000                    .WORD   0       ;15TH WORD CONTROL MAP..1920KW
734  002374  000000                    .WORD   0       ;16TH WORD CONTROL MAP..2048KW
735  002376  000000                    .WORD   0       ;MASK FOR UNUSED PARITY CSR BITS
736  002400  172105            MPR2:   172104 +1       ;PARITY STATUS REGISTER
737  002402  000000                    .WORD   0       ;1ST  WORD CONTROL MAP...128KW
738  002404  000000                    .WORD   0       ;2ND  WORD CONTROL MAP...256KW
739  002406  000000                    .WORD   0       ;3RD  WORD CONTROL MAP...384KW
740  002410  000000                    .WORD   0       ;4TH  WORD CONTROL MAP...512KW
741  002412  000000                    .WORD   0       ;5TH  WORD CONTROL MAP...640KW
742  002414  000000                    .WORD   0       ;6TH  WORD CONTROL MAP...768KW
743  002416  000000                    .WORD   0       ;7TH  WORD CONTROL MAP...896KW
744  002420  000000                    .WORD   0       ;8TH  WORD CONTROL MAP..1024KW
745  002422  000000                    .WORD   0       ;9TH  WORD CONTROL MAP..1152KW
746  002424  000000                    .WORD   0       ;10TH WORD CONTROL MAP..1280KW
747  002426  000000                    .WORD   0       ;11TH WORD CONTROL MAP..1408KW
748  002430  000000                    .WORD   0       ;12TH WORD CONTROL MAP..1536KW
749  002432  000000                    .WORD   0       ;13TH WORD CONTROL MAP..1664KW
750  002434  000000                    .WORD   0       ;14TH WORD CONTROL MAP..1792KW
751  002436  000000                    .WORD   0       ;15TH WORD CONTROL MAP..1920KW
752  002440  000000                    .WORD   0       ;16TH WORD CONTROL MAP..2048KW
753  002442  000000                    .WORD   0       ;MASK FOR UNUSED PARITY CSR BITS
754  002444  172107            MPR3:   172106 +1       ;PARITY STATUS REGISTER
755  002446  000000                    .WORD   0       ;1ST  WORD CONTROL MAP...128KW
756  002450  000000                    .WORD   0       ;2ND  WORD CONTROL MAP...256KW
757  002452  000000                    .WORD   0       ;3RD  WORD CONTROL MAP...384KW
758  002454  000000                    .WORD   0       ;4TH  WORD CONTROL MAP...512KW
759  002456  000000                    .WORD   0       ;5TH  WORD CONTROL MAP...640KW
760  002460  000000                    .WORD   0       ;6TH  WORD CONTROL MAP...768KW
761  002462  000000                    .WORD   0       ;7TH  WORD CONTROL MAP...896KW
762  002464  000000                    .WORD   0       ;8TH  WORD CONTROL MAP..1024KW
763  002466  000000                    .WORD   0       ;9TH  WORD CONTROL MAP..1152KW
764  002470  000000                    .WORD   0       ;10TH WORD CONTROL MAP..1280KW
765  002472  000000                    .WORD   0       ;11TH WORD CONTROL MAP..1408KW
766  002474  000000                    .WORD   0       ;12TH WORD CONTROL MAP..1536KW
767  002476  000000                    .WORD   0       ;13TH WORD CONTROL MAP..1664KW
768  002500  000000                    .WORD   0       ;14TH WORD CONTROL MAP..1792KW
769  002502  000000                    .WORD   0       ;15TH WORD CONTROL MAP..1920KW
770  002504  000000                    .WORD   0       ;16TH WORD CONTROL MAP..2048KW
```

```
  771  002506  000000C                    .WORD   0       ;MASK FOR UNUSED PARITY CSR BITS
  772  002510  172111            MPR4:   172110 +1       ;PARITY STAUS REGISTER
  773  002512  000000                    .WORD   0       ;1ST  WORD CONTROL MAP...128KW
  774  002514  000000                    .WORD   0       ;2ND  WORD CONTROL MAP...256KW
  775  002516  000000                    .WORD   0       ;3RD  WORD CONTROL MAP...384KW
  776  002520  000000                    .WORD   0       ;4TH  WORD CONTROL MAP...512KW
  777  002522  000000                    .WORD   0       ;5TH  WORD CONTROL MAP...640KW
  778  002524  000000                    .WORD   0       ;6TH  WORD CONTROL MAP...768KW
  779  002526  000000                    .WORD   0       ;7TH  WORD CONTROL MAP...896KW
  780  002530  000000                    .WORD   0       ;8TH  WORD CONTROL MAP..1024KW
  781  002532  000000                    .WORD   0       ;9TH  WORD CONTROL MAP..1152KW
  782  002534  000000                    .WORD   0       ;10TH WORD CONTROL MAP..1280KW
  783  002536  000000                    .WORD   0       ;11TH WORD CONTROL MAP..1408KW
  784  002540  000000                    .WORD   0       ;12TH WORD CONTROL MAP..1536KW
  785  002542  000000                    .WORD   0       ;13TH WORD CONTROL MAP..1664KW
  786  002544  000000                    .WORD   0       ;14TH WORD CONTROL MAP..1792KW
  787  002546  000000                    .WORD   0       ;15TH WORD CONTROL MAP..1920KW
  788  002550  000000                    .WORD   0       ;16TH WORD CONTROL MAP..2048KW
  789  002552  000000                    .WORD   0       ;MASK FOR UNUSED PARITY CSR BITS
  790  002554  172113            MPR5:   172112 +1       ;PARITY STATUS REGISTER
  791  002556  000000                    .WORD   0       ;1ST  WORD CONTROL MAP...128KW
  792  002560  000000                    .WORD   0       ;2ND  WORD CONTROL MAP...256KW
  793  002562  000000                    .WORD   0       ;3RD  WORD CONTROL MAP...384KW
  794  002564  000000                    .WORD   0       ;4TH  WORD CONTROL MAP...512KW
  795  002566  000000                    .WORD   0       ;5TH  WORD CONTROL MAP...640KW
  796  002570  000000                    .WORD   0       ;6TH  WORD CONTROL MAP...768KW
  797  002572  000000                    .WORD   0       ;7TH  WORD CONTROL MAP...896KW
  798  002574  000000                    .WORD   0       ;8TH  WORD CONTROL MAP..1024KW
  799  002576  000000                    .WORD   0       ;9TH  WORD CONTROL MAP..1152KW
  800  002600  000000                    .WORD   0       ;10TH WORD CONTROL MAP..1280KW
  801  002602  000000                    .WORD   0       ;11TH WORD CONTROL MAP..1408KW
  802  002604  000000                    .WORD   0       ;12TH WORD CONTROL MAP..1536KW
  803  002606  000000                    .WORD   0       ;13TH WORD CONTROL MAP..1664KW
  804  002610  000000                    .WORD   0       ;14TH WORD CONTROL MAP..1792KW
  805  002612  000000                    .WORD   0       ;15TH WORD CONTROL MAP..1920KW
  806  002614  000000                    .WORD   0       ;16TH WORD CONTROL MAP..2048KW
  807  002616  000000                    .WORD   0       ;MASK FOR UNUSED PARITY CSR BITS
  808  002620  172115            MPR6:   172114 +1       ;PARITY STATUS REGISTER
  809  002622  000000                    .WORD   0       ;1ST  WORD CONTROL MAP...128KW
  810  002624  000000                    .WORD   0       ;2ND  WORD CONTROL MAP...256KW
  811  002626  000000                    .WORD   0       ;3RD  WORD CONTROL MAP...384KW
  812  002630  000000                    .WORD   0       ;4TH  WORD CONTROL MAP...512KW
  813  002632  000000                    .WORD   0       ;5TH  WORD CONTROL MAP...640KW
  814  002634  000000                    .WORD   0       ;6TH  WORD CONTROL MAP...768KW
  815  002636  000000                    .WORD   0       ;7TH  WORD CONTROL MAP...896KW
  816  002640  000000                    .WORD   0       ;8TH  WORD CONTROL MAP..1024KW
  817  002642  000000                    .WORD   0       ;9TH  WORD CONTROL MAP..1152KW
  818  002644  000000                    .WORD   0       ;10TH WORD CONTROL MAP..1280KW
  819  002646  000000                    .WORD   0       ;11TH WORD CONTROL MAP..1408KW
  820  002650  000000                    .WORD   0       ;12TH WORD CONTROL MAP..1536KW
  821  002652  000000                    .WORD   0       ;13TH WORD CONTROL MAP..1664KW
  822  002654  000000                    .WORD   0       ;14TH WORD CONTROL MAP..1792KW
  823  002656  000000                    .WORD   0       ;15TH WORD CONTROL MAP..1920KW
  824  002660  000000                    .WORD   0       ;16TH WORD CONTROL MAP..2048KW
  825  002662  000000                    .WORD   0       ;MASK FOR UNUSED PARITY CSR BITS
  826  002664  172117            MPR7:   172116 +1       ;PARITY STATUS REGISTER
```

```
 827  002666  000000C                        .WORD   0       ;1ST  WORD CONTROL MAP...128KW
 828  002670  000000                         .WORD   0       ;2ND  WORD CONTROL MAP...256KW
 829  002672  000000                         .WORD   0       ;3RD  WORD CONTROL MAP...384KW
 830  002674  000000                         .WORD   0       ;4TH  WORD CONTROL MAP...512KW
 831  002676  000000                         .WORD   0       ;5TH  WORD CONTROL MAP...640KW
 832  002700  000000                         .WORD   0       ;6TH  WORD CONTROL MAP...768KW
 833  002702  000000                         .WORD   0       ;7TH  WORD CONTROL MAP...896KW
 834  002704  000000                         .WORD   0       ;8TH  WORD CONTROL MAP..1024KW
 835  002706  000000                         .WORD   0       ;9TH  WORD CONTROL MAP..1152KW
 836  002710  000000                         .WORD   0       ;10TH WORD CONTROL MAP..1280KW
 837  002712  000000                         .WORD   0       ;11TH WORD CONTROL MAP..1408KW
 838  002714  000000                         .WORD   0       ;12TH WORD CONTROL MAP..1536KW
 839  002716  000000                         .WORD   0       ;13TH WORD CONTROL MAP..1664KW
 840  002720  000000                         .WORD   0       ;14TH WORD CONTROL MAP..1792KW
 841  002722  000000                         .WORD   0       ;15TH WORD CONTROL MAP..1920KW
 842  002724  000000                         .WORD   0       ;16TH WORD CONTROL MAP..2048KW
 843  002726  000000                         .WORD   0       ;MASK FOR UNUSED PARITY CSR BITS
 844  002730  172121              MPR8:      172120 +1       ;PARITY STATUS REGISTER
 845  002732  000000                         .WORD   0       ;1ST  WORD CONTROL MAP...128KW
 846  002734  000000                         .WORD   0       ;2ND  WORD CONTROL MAP...256KW
 847  002736  000000                         .WORD   0       ;3RD  WORD CONTROL MAP...384KW
 848  002740  000000                         .WORD   0       ;4TH  WORD CONTROL MAP...512KW
 849  002742  000000                         .WORD   0       ;5TH  WORD CONTROL MAP...640KW
 850  002744  000000                         .WORD   0       ;6TH  WORD CONTROL MAP...768KW
 851  002746  000000                         .WORD   0       ;7TH  WORD CONTROL MAP...896KW
 852  002750  000000                         .WORD   0       ;8TH  WORD CONTROL MAP..1024KW
 853  002752  000000                         .WORD   0       ;9TH  WORD CONTROL MAP..1152KW
 854  002754  000000                         .WORD   0       ;10TH WORD CONTROL MAP..1280KW
 855  002756  000000                         .WORD   0       ;11TH WORD CONTROL MAP..1408KW
 856  002760  000000                         .WORD   0       ;12TH WORD CONTROL MAP..1536KW
 857  002762  000000                         .WORD   0       ;13TH WORD CONTROL MAP..1664KW
 858  002764  000000                         .WORD   0       ;14TH WORD CONTROL MAP..1792KW
 859  002766  000000                         .WORD   0       ;15TH WORD CONTROL MAP..1920KW
 860  002770  000000                         .WORD   0       ;16TH WORD CONTROL MAP..2048KW
 861  002772  000000                         .WORD   0       ;MASK FOR UNUSED PARITY CSR BITS
 862  002774  172123              MPR9:      172122 +1       ;PARITY STATUS REGISTER
 863  002776  000000                         .WORD   0       ;1ST  WORD CONTROL MAP...128KW
 864  003000  000000                         .WORD   0       ;2ND  WORD CONTROL MAP...256KW
 865  003002  000000                         .WORD   0       ;3RD  WORD CONTROL MAP...384KW
 866  003004  000000                         .WORD   0       ;4TH  WORD CONTROL MAP...512KW
 867  003006  000000                         .WORD   0       ;5TH  WORD CONTROL MAP...640KW
 868  003010  000000                         .WORD   0       ;6TH  WORD CONTROL MAP...768KW
 869  003012  000000                         .WORD   0       ;7TH  WORD CONTROL MAP...896KW
 870  003014  000000                         .WORD   0       ;8TH  WORD CONTROL MAP..1024KW
 871  003016  000000                         .WORD   0       ;9TH  WORD CONTROL MAP..1152KW
 872  003020  000000                         .WORD   0       ;10TH WORD CONTROL MAP..1280KW
 873  003022  000000                         .WORD   0       ;11TH WORD CONTROL MAP..1408KW
 874  003024  000000                         .WORD   0       ;12TH WORD CONTROL MAP..1536KW
 875  003026  000000                         .WORD   0       ;13TH WORD CONTROL MAP..1664KW
 876  003030  000000                         .WORD   0       ;14TH WORD CONTROL MAP..1792KW
 877  003032  000000                         .WORD   0       ;15TH WORD CONTROL MAP..1920KW
 878  003034  000000                         .WORD   0       ;16TH WORD CONTROL MAP..2048KW
 879  003036  000000                         .WORD   0       ;MASK FOR UNUSED PARITY CSR BITS
 880  003040  172125              MPR10:     172124 +1       ;PARITY STATUS REGISTER
 881  003042  000000                         .WORD   0       ;1ST  WORD CONTROL MAP...128KW
 882  003044  000000                         .WORD   0       ;2ND  WORD CONTROL MAP...256KW
```

```
 883  003046  000000                    .WORD   0       ;3RD  WORD CONTROL MAP...384KW
 884  003050  000000                    .WORD   0       ;4TH  WORD CONTROL MAP...512KW
 885  003052  000000                    .WORD   0       ;5TH  WORD CONTROL MAP...640KW
 886  003054  000000                    .WORD   0       ;6TH  WORD CONTROL MAP...768KW
 887  003056  000000                    .WORD   0       ;7TH  WORD CONTROL MAP...896KW
 888  003060  000000                    .WORD   0       ;8TH  WORD CONTROL MAP..1024KW
 889  003062  000000                    .WORD   0       ;9TH  WORD CONTROL MAP..1152KW
 890  003064  000000                    .WORD   0       ;10TH WORD CONTROL MAP..1280KW
 891  003066  000000                    .WORD   0       ;11TH WORD CONTROL MAP..1408KW
 892  003070  000000                    .WORD   0       ;12TH WORD CONTROL MAP..1536KW
 893  003072  000000                    .WORD   0       ;13TH WORD CONTROL MAP..1664KW
 894  003074  000000                    .WORD   0       ;14TH WORD CONTROL MAP..1792KW
 895  003076  000000                    .WORD   0       ;15TH WORD CONTROL MAP..1920KW
 896  003100  000000                    .WORD   0       ;16TH WORD CONTROL MAP..2048KW
 897  003102  000000                    .WORD   0       ;MASK FOR UNUSED PARITY CSR BITS
 898  003104  172127          MPR11:    172126 +1       ;PARITY STATUS REGISTER
 899  003106  000000                    .WORD   0       ;1ST  WORD CONTROL MAP...128KW
 900  003110  000000                    .WORD   0       ;2ND  WORD CONTROL MAP...256KW
 901  003112  000000                    .WORD   0       ;3RD  WORD CONTROL MAP...384KW
 902  003114  000000                    .WORD   0       ;4TH  WORD CONTROL MAP...512KW
 903  003116  000000                    .WORD   0       ;5TH  WORD CONTROL MAP...640KW
 904  003120  000000                    .WORD   0       ;6TH  WORD CONTROL MAP...768KW
 905  003122  000000                    .WORD   0       ;7TH  WORD CONTROL MAP...896KW
 906  003124  000000                    .WORD   0       ;8TH  WORD CONTROL MAP..1024KW
 907  003126  000000                    .WORD   0       ;9TH  WORD CONTROL MAP..1152KW
 908  003130  000000                    .WORD   0       ;10TH WORD CONTROL MAP..1280KW
 909  003132  000000                    .WORD   0       ;11TH WORD CONTROL MAP..1408KW
 910  003134  000000                    .WORD   0       ;12TH WORD CONTROL MAP..1536KW
 911  003136  000000                    .WORD   0       ;13TH WORD CONTROL MAP..1664KW
 912  003140  000000                    .WORD   0       ;14TH WORD CONTROL MAP..1792KW
 913  003142  000000                    .WORD   0       ;15TH WORD CONTROL MAP..1920KW
 914  003144  000000                    .WORD   0       ;16TH WORD CONTROL MAP..2048KW
 915  003146  000000                    .WORD   0       ;MASK FOR UNUSED PARITY CSR BITS
 916  003150  172131          MPR12:    172130 +1       ;PARITY STATUS REGISTER
 917  003152  000000                    .WORD   0       ;1ST  WORD CONTROL MAP...128KW
 918  003154  000000                    .WORD   0       ;2ND  WORD CONTROL MAP...256KW
 919  003156  000000                    .WORD   0       ;3RD  WORD CONTROL MAP...384KW
 920  003160  000000                    .WORD   0       ;4TH  WORD CONTROL MAP...512KW
 921  003162  000000                    .WORD   0       ;5TH  WORD CONTROL MAP...640KW
 922  003164  000000                    .WORD   0       ;6TH  WORD CONTROL MAP...768KW
 923  003166  000000                    .WORD   0       ;7TH  WORD CONTROL MAP...896KW
 924  003170  000000                    .WORD   0       ;8TH  WORD CONTROL MAP..1024KW
 925  003172  000000                    .WORD   0       ;9TH  WORD CONTROL MAP..1152KW
 926  003174  000000                    .WORD   0       ;10TH WORD CONTROL MAP..1280KW
 927  003176  000000                    .WORD   0       ;11TH WORD CONTROL MAP..1408KW
 928  003200  000000                    .WORD   0       ;12TH WORD CONTROL MAP..1536KW
 929  003202  000000                    .WORD   0       ;13TH WORD CONTROL MAP..1664KW
 930  003204  000000                    .WORD   0       ;14TH WORD CONTROL MAP..1792KW
 931  003206  000000                    .WORD   0       ;15TH WORD CONTROL MAP..1920KW
 932  003210  000000                    .WORD   0       ;16TH WORD CONTROL MAP..2048KW
 933  003212  000000                    .WORD   0       ;MASK FOR UNUSED PARITY CSR BITS
 934  003214  172133          MPR13:    172132 +1       ;PARITY STATUS REGISTER
 935  003216  000000                    .WORD   0       ;1ST  WORD CONTROL MAP...128KW
 936  003220  000000                    .WORD   0       ;2ND  WORD CONTROL MAP...256KW
 937  003222  000000                    .WORD   0       ;3RD  WORD CONTROL MAP...384KW
 938  003224  000000                    .WORD   0       ;4TH  WORD CONTROL MAP...512KW
```

```
 939  003226  000000                    .WORD   0       ;5TH  WORD CONTROL MAP...640KW
 940  003230  000000                    .WORD   0       ;6TH  WORD CONTROL MAP...768KW
 941  003232  000000                    .WORD   0       ;7TH  WORD CONTROL MAP...896KW
 942  003234  000000                    .WORD   0       ;8TH  WORD CONTROL MAP..1024KW
 943  003236  000000                    .WORD   0       ;9TH  WORD CONTROL MAP..1152KW
 944  003240  000000                    .WORD   0       ;10TH WORD CONTROL MAP..1280KW
 945  003242  000000                    .WORD   0       ;11TH WORD CONTROL MAP..1408KW
 946  003244  000000                    .WORD   0       ;12TH WORD CONTROL MAP..1536KW
 947  003246  000000                    .WORD   0       ;13TH WORD CONTROL MAP..1664KW
 948  003250  000000                    .WORD   0       ;14TH WORD CONTROL MAP..1792KW
 949  003252  000000                    .WORD   0       ;15TH WORD CONTROL MAP..1920KW
 950  003254  000000                    .WORD   0       ;16TH WORD CONTROL MAP..2048KW
 951  003256  000000                    .WORD   0       ;MASK FOR UNUSED PARITY CSR BITS
 952  003260  172135          MPR14:    172134 +1       ;PARITY STATUS REGISTER
 953  003262  000000                    .WORD   0       ;1ST  WORD CONTROL MAP...128KW
 954  003264  000000                    .WORD   0       ;2ND  WORD CONTROL MAP...256KW
 955  003266  000000                    .WORD   0       ;3RD  WORD CONTROL MAP...384KW
 956  003270  000000                    .WORD   0       ;4TH  WORD CONTROL MAP...512KW
 957  003272  000000                    .WORD   0       ;5TH  WORD CONTROL MAP...640KW
 958  003274  000000                    .WORD   0       ;6TH  WORD CONTROL MAP...768KW
 959  003276  000000                    .WORD   0       ;7TH  WORD CONTROL MAP...896KW
 960  003300  000000                    .WORD   0       ;8TH  WORD CONTROL MAP..1024KW
 961  003302  000000                    .WORD   0       ;9TH  WORD CONTROL MAP..1152KW
 962  003304  000000                    .WORD   0       ;10TH WORD CONTROL MAP..1280KW
 963  003306  000000                    .WORD   0       ;11TH WORD CONTROL MAP..1408KW
 964  003310  000000                    .WORD   0       ;12TH WORD CONTROL MAP..1536KW
 965  003312  000000                    .WORD   0       ;13TH WORD CONTROL MAP..1664KW
 966  003314  000000                    .WORD   0       ;14TH WORD CONTROL MAP..1792KW
 967  003316  000000                    .WORD   0       ;15TH WORD CONTROL MAP..1920KW
 968  003320  000000                    .WORD   0       ;16TH WORD CONTROL MAP..2048KW
 969  003322  000000                    .WORD   0       ;MASK FOR UNUSED PARITY CSR BITS
 970  003324  172137          MPR15:    172136 +1       ;PARITY STATUS REGISTER
 971  003326  000000                    .WORD   0       ;1ST  WORD CONTROL MAP...128KW
 972  003330  000000                    .WORD   0       ;2ND  WORD CONTROL MAP...256KW
 973  003332  000000                    .WORD   0       ;3RD  WORD CONTROL MAP...384KW
 974  003334  000000                    .WORD   0       ;4TH  WORD CONTROL MAP...512KW
 975  003336  000000                    .WORD   0       ;5TH  WORD CONTROL MAP...640KW
 976  003340  000000                    .WORD   0       ;6TH  WORD CONTROL MAP...768KW
 977  003342  000000                    .WORD   0       ;7TH  WORD CONTROL MAP...896KW
 978  003344  000000                    .WORD   0       ;8TH  WORD CONTROL MAP..1024KW
 979  003346  000000                    .WORD   0       ;9TH  WORD CONTROL MAP..1152KW
 980  003350  000000                    .WORD   0       ;10TH WORD CONTROL MAP..1280KW
 981  003352  000000                    .WORD   0       ;11TH WORD CONTROL MAP..1408KW
 982  003354  000000                    .WORD   0       ;12TH WORD CONTROL MAP..1536KW
 983  003356  000000                    .WORD   0       ;13TH WORD CONTROL MAP..1664KW
 984  003360  000000                    .WORD   0       ;14TH WORD CONTROL MAP..1792KW
 985  003362  000000                    .WORD   0       ;15TH WORD CONTROL MAP..1920KW
 986  003364  000000                    .WORD   0       ;16TH WORD CONTROL MAP..2048KW
 987  003366  000000                    .WORD   0               ;MASK FOR UNUSED PARITY CSR BITS
 988                      ;THIS IS THE END OF THE TABLE !
 989  003370  000021          MPRX:     .BLKW   17.     ;TABLE TO HOLD JUST PARITY STATUS REGISTERS THAT EXIST.
 990                                                   ; (THE EXTRA WORD IS FOR A TERMINATOR.)
 991
```

```
                                .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


003432                          $ERRTB:
003432                          CHGG1:
                                ;*  ITEM 1
003432  024774                      DM1             ;PARITY REGISTER DATA ERROR.
003434  026334                      DH1             ;PC,REG,S/B,WAS
003436  002040                      DT1             ;$ERRPC,$GDADR,$GDDAT,$BDDAT
003440  027072                      DF1             ;16,22,16,16
                                ;*  ITEM 2
003442  025030                      DM2             ;ADDRESS TEST ERROR(TST1-5).
003444  026357                      DH2             ;V/PC,P/PC,MA,S/B,WAS

003446  002052                      DT2             ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
003450  027076                      DF2             ;16,22,22,16,16
                                ;*  ITEM 3
003452  025030                      DM2             ;ADDRESS TEST ERROR(TST1-5).
003454  027036                      DH31            ;V/PC,P/PC,MA,S/B,WAS
003456  002052                      DT2             ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
003460  027103                      DF3             ;16,22,22,8,8
                                ;*  ITEM 4
003462  025064                      DM4             ;CONSTANT DATA ERROR(TST6-10).
003464  026357                      DH2             ;V/PC,P/PC,MA,S/B,WAS
003466  002052                      DT2             ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
003470  027076                      DF2             ;16,22,22,16,16
                                ;*  ITEM 5
003472  025122                      DM5             ;ROTATING BIT ERROR(TST11-12).
003474  026357                      DH2             ;V/PC,P/PC,MA,S/B,WAS
003476  002052                      DT2             ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
003500  027076                      DF2             ;16,22,22,16,16
                                ;*  ITEM 6
003502  025160                      DM6             ;MOS REFRESH TEST ERROR (TST24-25).
003504  026357                      DH2             ;V/PC,P/PC,MA,S/B,WAS
003506  002052                      DT2             ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
003510  027076                      DF2             ;16,22,22,16,16
                                ;*  ITEM 7
003512  025224                      DM7             ;FATAL ERROR HALT
003514  000000                      0
003516  000000                      0
003520  000000                      0
                                ;*  ITEM 10
003522  025245                      DM10            ;MARCHING 1'S AND 0'S ERROR(TST23).
003524  026357                      DH2             ;V/PC,P/PC,MA,S/B,WAS
003526  002052                      DT2             ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
```

```
1048  003530  027076                    DF2             ;16,22,22,16,16
1049                              ;*  ITEM 11
1050  003532  025311                    DM11            ;PARITY MEMORY ADDRESS ERROR(TST13).
1051  003534  027036                    DH31            ;V/PC,P/PC,MA,S/B,WAS
1052  003536  002052                    DT2             ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
1053  003540  027103                    DF3             ;16,22,22,8,8
1054                              ;*  ITEM 12
1055  003542  025355                    DM12            ;DATIO WITH WRONG PARITY DIDN'T TRAP(TST13).
1056  003544  026416                    DH12            ;V/PC,P/PC,MA,S/B
1057  003546  002066                    DT12            ;$VERPC,$ERRPC,$GDADR,$GDDAT
1058  003550  027103                    DF3             ;16,22,22,8
1059                              ;*  ITEM 13
1060  003552  025431                    DM13            ;WRONG PARITY DETECTED, BUT NO REGISTER SHOWS ERROR FLAG.
1061  003554  026416                    DH12            ;V/PC,P/PC,MA,S/B
1062  003556  002066                    DT12            ;$VERPC,$ERRPC,$GDADR,$GDDAT
1063  003560  027103                    DF3             ;16,22,22,8
1064                              ;*  ITEM 14
1065  003562  025522                    DM14            ;PARITY REGISTER NOT MAPPED AS CONTROLLING THIS ADDRESS(TST13).
1066  003564  026445                    DH14            ;V/PC,P/PC,REG,MA
1067  003566  002100                    DT14            ;$VERPC,$ERRPC,$TMP0,$GDADR
1068
1069  003570  027110                    DF14            ;16,22,22,22
1070                              ;*  ITEM 15
1071  003572  024774                    DM1             ;PARITY REGISTER DATA ERROR.
1072  003574  026474                    DH15            ;V/PC,P/PC,MAUT,REG,S/B,WAS
1073  003576  002112                    DT15            ;$VERPC,$ERRPC,$GDADR,$TMP0,$GDDAT,$BDDAT
1074  003600  027110                    DF14            ;16,22,22,22,16,16
1075                              ;*  ITEM 16
1076  003602  025621                    DM16            ;MORE THAN ONE REGISTER INDICATED PARITY ERROR.
1077  003604  026445                    DH14            ;V/PC,P/PC,REG,MA
1078  003606  002100                    DT14            ;$VERPC,$ERRPC,$TMP0,$GDADR
1079  003610  027110                    DF14            ;16,22,22,22
1080                              ;*  ITEM 17
1081  003612  025700                    DM17            ;DATA SHOULDN'T HAVE CHANGED WHEN PARITY ERROR
1082                                                    ; TRAPPED(TST13).
1083  003614  027036                    DH31            ;V/PC,P/PC,MA,S/B,WAS
1084  003616  002052                    DT2             ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
1085  003620  027103                    DF3             ;16,22,22,8,8
1086                              ;*  ITEM 20
1087  003622  025776                    DM20            ;RANDOM DATA ERROR(TST14).
1088  003624  026357                    DH2             ;V/PC,P/PC,MA,S/B,WAS
1089  003626  002052                    DT2             ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
1090  003630  027076                    DF2             ;16,22,22,16,16
1091                              ;*  ITEM 21
1092  003632  026030                    DM21            ;INSTRUCTION EXECUTION ERROR(TST15-22).
1093  003634  026547                    DH21            ;V/PC,P/PC,IUT,MA,S/B,WAS
1094  003636  002130                    DT21            ;$VERPC,$ERRPC,$TMP0,$GDADR,$GDDAT,$BDDAT
1095  003640  027116                    DF21            ;16,22,16,22,16,16
1096                              ;*  ITEM 22         ;NOT USED
1097  003642  000000                    0               ;CHGG1
1098  003644  000000                    0
1099  003646  000000                    0
1100  003650  000000                    0
1101                              ;*  ITEM 23
1102  003652  026077                    DM23            ;PROGRAM CODE CHANGED WHEN RELOCATED.
1103  003654  026614                    DH23            ;V/PC,P/PC,SRC MA,DST MA,S/B,WAS
```

```
1104  003656  002146                          DT23            ;$VERPC,$ERRPC,$GDADR,$BDADR,$GDDAT,$BDDAT
1105  003660  027110                          DF14            ;16,22,22,22,16,16
1106                                  ;*  ITEM 24
1107  003662  026144                          DM24            ;TRAPPED, BUT NO REGISTER HAD ERROR BIT SET.
1108  003664  026670                          DH24            ;V/PC,P/PC,TRP/PC
1109  003666  002164                          DT24            ;$VERPC,$ERRPC,$BDADR
1110  003670  027110                          DF14            ;16,22,22
1111                                  ;*  ITEM 25
1112  003672  026220                          DM25            ;TRAPPED TO 114.
1113  003674  026713                          DH25            ;V/PC,P/PC,TRP/PC,REG,WAS
1114  003676  002174                          DT25            ;$VERPC,$ERRPC,$BDADR,$TMP0,$TMP1
1115  003700  027110                          DF14            ;16,22,22,22,16
1116                                  ;*  ITEM 26
1117  003702  026240                          DM26            ;FAILED TO TRAP.
1118  003704  026757                          DH26            ;V/PC,P/PC,REG,WAS
1119  003706  002210                          DT26            ;$VERPC,$ERRPC,$TMP0,$TMP1
1120  003710  027076                          DF2             ;16,22,22,16
1121                                  ;*  ITEM 27
1122  003712  026260                          DM27            ;(ACTION ENABLE WASN'T SET).
1123  003714  026757                          DH26            ;V/PC,P/PC,REG,WAS
1124  003716  002210                          DT26            ;$VERPC,$ERRPC,$TMP0,$TMP1
1125  003720  027076                          DF2             ;16,22,22,16
1126                                  ;*  ITEM 30
1127  003722  000000                          0               ;NO MESSAGE.
1128  003724  027007                          DH30            ;REG,WAS,MA,WAS
1129  003726  002222                          DT30            ;$TMP0,$TMP1,$GDADR,$BDDAT
1130  003730  027124                          DF30            ;22,16,22,8
1131                                  ;*  ITEM 31
1132  003732  026314                          DM31            ;TRAPPED TO 4
1133  003734  000000                          0               ;NO HEADER
1134  003736  002234                          DT31            ;$TMP3
1135  003740  027124                          DF30            ;22
1136
1137                                  .SBTTL START:   SETUP AND MAP MEMORY

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

1144  003742  105067  176010          START:  CLRB    SELFLG          ;CLEAR SELECT PARAMETER FLAG.
1145  003746  000403                          BR      STARTA          ;GO DO SETUP AND MEMORY MAP.
1146  003750  112767  177777  176000  SELECT: MOVB    #-1,    SELFLG  ;SET THE SELECT PARAMETERS FLAG.
1147  003756                          STARTA:
1148                                  .SBTTL  INITIALIZE THE COMMON TAGS
1149                                  ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
1150  003756  012706  001070                  MOV     #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
1151  003762  005026                  10$:    CLR     (R6)+           ;;CLEAR MEMORY LOCATION
1152  003764  022706  001130                  CMP     #SWR,R6 ;;DONE?
1153  003770  001374                          BNE     .-6             ;;LOOP BACK IF NO
1154  003772  012706  001100                  MOV     #STACK,SP       ;;SETUP THE STACK POINTER
1155                                  ;;INITIALIZE A FEW VECTORS
1156  003776  012737  000540  000024          MOV     #$PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
1157  004004  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;LEVEL 7
1158  004012  016767  006474  006464          MOV     $ENDCT,$EOPCT   ;;SETUP END-OF-PROGRAM COUNTER
1159                                  ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
```

```
                                        ;;EQUAL TO A ''-1'', SETUP FOR A SOFTWARE SWITCH REGISTER.
004020  013746  000004                    MOV     @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
004024  012737  004060  000004            MOV     #64$,@#ERRVEC   ;;SET UP ERROR VECTOR
004032  012767  177570  175070            MOV     #DSWR,SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
004040  012767  177570  175064            MOV     #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
004046  022777  177777  175054            CMP     #-1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
004054  001012                            BNE     66$             ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
                                                                ;;AND  THE HARDWARE SWR IS NOT = -1
004056  000403                            BR      65$             ;;BRANCH IF NO TIMEOUT
004060  012716  004066            64$:    MOV     #65$,(SP)       ;;SET UP FOR TRAP RETURN
004064  000002                            RTI
004066  012767  000176  175034    65$:    MOV     #SWREG,SWR      ;;POINT TO SOFTWARE SWR
004074  012767  000174  175030            MOV     #DISPREG,DISPLAY
004102  012637  000004            66$:    MOV     (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR

004106  005067  175070                    CLR     $PASS           ;;CLEAR PASS COUNT
004112  132767  000200  175075            BITB    #APTSIZE,$ENVM  ;;TEST USER SIZE UNDER APT
004120  001403                            BEQ     67$             ;;YES,USE NON-APT SWITCH
004122  012767  001216  175000            MOV     #$SWREG,SWR     ;;NO,USE APT SWITCH REGISTER
004130                              67$:
004130  005067  175356                    CLR     LDDISP          ;CLEAR DISPLAY REGISTER STORAGE LOCN
004134  005077  174772                    CLR     @DISPLAY        ;CLEAR DISPLAY REGISTER
                                        .SBTTL  TYPE PROGRAM NAME
                                        ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
004140  005227  177777                    INC     #-1             ;;FIRST TIME?
004144  001041                            BNE     68$             ;;BRANCH IF NO
004146  022737  012560  000042            CMP     #$ENDAD,@#42    ;;ACT-11?
004154  001435                            BEQ     68$             ;;BRANCH IF YES
004156  004567  015412                    JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
004162  004236                            .WORD   69$             ;ADDRESS OF MESSAGE TO BE TYPED
                                        .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
004164  005737  000042                    TST     @#42            ;;ARE WE RUNNING UNDER XXDP/ACT?
004170  001016                            BNE     70$             ;;BRANCH IF YES
004172  126727  175016  000001            CMPB    $ENV,#1         ;;ARE WE RUNNING UNDER APT?
004200  001412                            BEQ     70$             ;;BRANCH IF YES
004202  026727  174722  000176            CMP     SWR,#SWREG      ;;SOFTWARE SWITCH REG SELECTED?
004210  001011                            BNE     71$             ;;BRANCH IF NO
                                        ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $GTSWR ROUTINE
                                        ;*  WIHTOUT USING A ''TRAP'' INSTRUCTION AS CALLED FOR BY **SYSMAC**.
004212  106746                            MFPS    -(SP)           ;PUT THE PROCESSOR STATUS ON THE STACK
004214  105066  000001                    CLRB    1(SP)           ;HIGH BYTE CLEARED TO INSURE KERNEL MODE
                                                                ;ON PSW RETURN.
004220  004767  014256                    JSR     PC,     $GTSWR  ;GO TO THE SUBROUTINE
004224  000403                            BR      71$
004226  112767  000001  174670    70$:    MOVB    #1,$AUTOB       ;;SET AUTO-MODE INDICATOR
004234                              71$:
004234  000405                            BR      68$             ;;GET OVER THE ASCIZ
                                        ;;69$:  .ASCIZ  <CRLF>'CVMSAA'<CRLF>
004250                              68$:
004250  010700                            MOV     PC,     R0      ;GET CURRENT PROGRAM COUNTER.
004252  022700  004252            9$:     CMP     #9$,    R0      ;CHECK IF THE PROGRAM IS RELOCATED.
004256  001402                            BEQ     10$             ;BR IF PROGRAM NOT RELOCATED.
004260  000167  174014                    JMP     RESTAR          ;GO TRY TO RELOCTED BEFORE CONTINUING.
004264  012767  000001  174242    10$:    MOV     #1,     PRGMAP  ;INITIALIZE PROGRAM MAP.
004272  005067  174234            12$:    CLR     RELOCF          ;INIT THE RELOCATION FACTOR.
004276  105737  001214                    TSTB    @#$ENV          ;CHECK FOR APT11
```

```
1216  004302  001011                        BNE     13$             ;BR IF APT11
1217  004304  005737  000042                TST     @#42            ;CHECK FOR STANDALONE
1218  004310  001406                        BEQ     13$             ;BR IF STANDALONE
1219  004312  023737  000042  000046        CMP     @#42,@#46       ;CHECK FOR ACT11
1220  004320  001402                        BEQ     13$             ;BR IF ACT11
1221                                                                ;MUST BE XXDP
1222  004322  004767  011046                JSR     PC,SAVLDR       ;GO SAVE LOADERS
1223
1224                                ;*  CHECK IF MEMORY MANAGEMENT IS AVAILABLE, AND SET IT UP IF IT IS.
1225                                ;*  INITIALIZE THE MEMMAP TABLE
1226  004326  012700  001514        13$:    MOV     #MEMMAP,R0      ;LOAD R0 WITH MEMMAP TABLE ADDRESS
1227  004332  012701  000020                MOV     #16.,R1         ;LOAD COUNTER
1228  004336  005020                14$:    CLR     (R0)+           ;CLEAR MEMMAP TABLE ENTRY
1229  004340  077102                        SOB     R1,14$          ;DECREMENT COUNTER 16. TIMES
1230                                                                ;IF COUNTER NOT = 0 THEN CLEAR NEXT ENTRY
1231
1232  004342  005067  175430                CLR     LSIFLG          ;INIT LSI-11 /2/QUAD PROCESSOR FLAG
1233  004346  012767  004360  173434        MOV     #15$,RESVEC     ;FIND OUT IF LSI-11/2/QUAD
1234  004354  000007                        MFPT                    ;MFPT INSTRUCTION WILL CAUSE TRAP
1235                                                                ;ON LSI-11/2
1236  004356  000404                        BR      16$             ;11/23 OR LATER WILL BRANCH
1237  004360  062706  000004        15$:    ADD     #4,SP           ;LSI-11 RETURN,CORRECT STACK
1238  004364  005267  175406                INC     LSIFLG          ;AND SET LSI-11/2 FLAG
1239
1240                                ;*  CHECK IF MEMORY MANAGEMENT IS AVAILABLE, AND SET IT UP IF IT IS
1241  004370  005067  174142        16$:    CLR     MMAVA           ;CLEAR KT AND 22 BIT ADDRESSING FLAG
1242  004374  032777  010000  174526        BIT     #SW12,@SWR      ;CHECK FOR INHIBIT KT11 SWITCH
1243  004402  001047                        BNE     NONKT           ;BRANCH IF SET
1244  004404  012737  004522  000004        MOV     #NONKT,@#ERRVEC ;SET UP TIMEOUT TRAP VECTOR
1245  004412  005037  177572                CLR     @#SR0           ;CLEAR MEM MGMT STATUS REGISTER
1246  004416  004767  006214                JSR     PC,MMINIT       ;MEM MGMT INITIALIZATION ROUTINE
1247  004422  005267  174110                INC     MMAVA           ;SET MEM MGMT AVAILABLE FLAG
1248  004426  004567  015142                JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
1249  004432  023472                        .WORD   MMAMES          ;ADDRESS OF MESSAGE TO BE TYPED
1250                                                                ;'KT11 AVAILABLE''
1251
1252
1253                                ;*  CHECK IF 22 BIT SYSTEM AVAILABLE AND SET IT UP IF IT IS
1254  004434  012737  004504  000004        MOV     #22$,@#ERRVEC   ;SET UP FOR TIME OUT VECTOR
1255  004442  005037  000000                CLR     @#0             ;CLEAR LOCATION 0
1256  004446  012737  010000  172344        MOV     #10000,@#KIPAR2 ;SET PAR2 TO LOC 128K + 2
1257  004454  052737  000020  172516        BIS     #BIT4,@#SR3     ;TURN ON 22 BIT ADDRESSING
1258  004462  012737  177777  040000        MOV     #-1,@#40000     ;NOW WRITE TO LOC 128K + 2
1259  004470  005737  000000                TST     @#0             ;IF LOC 0 = 0
1260  004474  001403                        BEQ     22$             ;  THEN 22 BIT SYSTEM
1261  004476  005037  172516                CLR     @#SR3           ;  ELSE 18 BIT SYSTEM, DISABLE 22 BIT ADR
1262  004502  000454                        BR      KTSIZ           ;       AND GO SIZE MEMORY
1263
1264                                ;*  TIME OUT TRAP TO HERE OR MEMORY EXISTS AT 128K + 2
1265  004504  052767  100000  174024  22$:    BIS     #BIT15,MMAVA    ;  ELSE SET 22 BIT FLAG
1266  004512  004567  015056                JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
1267  004516  023537                        .WORD   AVAL22          ;ADDRESS OF MESSAGE TO BE TYPED
1268                                                                ;'22 BIT ADR AVAILABLE''
1269  004520  000445                        BR      KTSIZ           ;GO SIZE MEMORY
1270
1271
```

```
 1272                                   ;;*****************************************************************
 1273                                   ;*  THIS ROUTINE WILL MAP MEMORY IN 8K SEGMENTS.  SUPPORTS ONLY THE SIZING
 1274                                   ;*  OF 16 BIT ADDRESSING  WITHOUT MEM MGMT SUPPORT.
 1275                                   ;*      STORAGE USED:
 1276                                   ;*              R0      = MEMMAP POINTER ... LO 128K
 1277                                   ;*              R2      = ADDRESS POINTER
 1278                                   ;*              R3      = BANK POINTER ... LO 128K
 1279                                   ;*              FLG30K  = 30K MEMORY FLAG
 1280                                   ;*      LITERALS:
 1281                                   ;*              MASK8K  - 37777
 1282                                   ;;*****************************************************************
 1283  004522  012706  001100          NONKT:  MOV     #STACK,SP       ;SET-UP THE STACK
 1284  004526  012700  001514                  MOV     #MEMMAP,R0      ;SET UP MEMORY MAP PTR TO LO 128K
 1285  004532  005002                          CLR     R2              ;SET ADDRESS PTR TO 0
 1286  004534  012703  000001                  MOV     #1,R3           ;SET UP 8K BANK POINTER
 1287  004540  012737  004562  000004          MOV     #2$,@#ERRVEC    ;SET UP TIME OUT VECTOR
 1288  004546  011222                  1$:     MOV     (R2),(R2)+      ;READ AND WRITE ALL MEMORY
 1289  004550  032702  037777                  BIT     #MASK8K,R2      ;IF NOT 8K BOUNDARY
 1290  004554  001374                          BNE     1$              ;  THEN CHECK NEXT LOCATION
 1291  004556  050310                          BIS     R3,(R0)         ;  ELSE SET BANK FLAG IN MEMMAP
 1292  004560  000420                          BR      3$              ;       AND DO SOME MORE
 1293
 1294                                   ;*  TIMEOUT TRAPS TO HERE
 1295  004562  062706  000004          2$:     ADD     #4,SP           ;RESTORE STACK POINTER
 1296  004566  022702  160000                  CMP     #160000,R2      ;IF NOT 28K BOUNDARY
 1297  004572  001001                          BNE     20$             ;  THEN BRANCH
 1298  004574  000405                          BR      21$             ;  ELSE SET UP POINTERS
 1299  004576  022702  170000          20$:    CMP     #170000,R2      ;IF NOT 30K BOUNDARY
 1300  004602  001004                          BNE     22$             ;  THEN BRANCH
 1301  004604  005267  175164                  INC     FLG30K          ;  ELSE SET 30K MEMORY FLAG
 1302  004610  050310                  21$:    BIS     R3,(R0)         ;SET BANK FLAG IN MEMMAP
 1303  004612  000407                          BR      4$              ;BRANCH ALL DONE
 1304  004614  052702  037777          22$:    BIS     #MASK8K,R2      ;POINT TO LAST ADDRESS OF 8K BANK
 1305  004620  005202                          INC     R2              ;POINT TO 1ST ADDRESS OF NEXT BANK
 1306
 1307  004622  106303                  3$:     ASLB    R3              ;UPDATE BANK POINTER
 1308  004624  032703  000020                  BIT     #BIT4,R3        ;IF NOT DONE WITH 32K
 1309  004630  001746                          BEQ     1$              ;  THEN TRY SOME MORE
 1310  004632  000500                  4$:     BR      DISMAP          ;GO TYPE OUT MAP
 1311
 1312
 1313                                   ;;*****************************************************************
 1314                                   ;*  THIS ROUTINE WILL MAP MEMORY IN 8K SEGMENTS.  MEMORY MANAGEMENT REGISTERS
 1315                                   ;*  KIPAR2 AND KIPAR3 ARE USED TO MAP THE 8K BANKS OF MEMORY.
 1316                                   ;*  IF MEMORY EXISTS NEXT TO THE I/O PAGE (I.E. 760000 OR 17760000)
 1317                                   ;*  THEN THE LAST BANK WILL BE ACKNOWLEDGED AS EXISTING.
 1318                                   ;*      STORAGE USED:
 1319                                   ;*              R0      = MEMMAP POINTER
 1320                                   ;*              R2      = ADDRESS POINTER
 1321                                   ;*              R3      = BANK POINTER
 1322                                   ;*              KIPAR2  = MAPPED TO 1ST 4K OF PRESENT 8K BANK
 1323                                   ;*              KIPAR3  = MAPPED TO 2ND 4K OF PRESENT 8K BANK
 1324                                   ;*      LITERALS USED:
 1325                                   ;*              MASK8K  = MASK OF 8K (37777)
 1326                                   ;;*****************************************************************
 1327  004634  012706  001100          KTSIZ:  MOV     #STACK,SP       ;SET-UP THE STACK
```

```
1328  004640  012700  001514            MOV     #MEMMAP,R0      ;SET-UP MEMMAP PTR TO FIRST ENTRY
1329  004644  012702  040000            MOV     #40000,R2       ;INIT VIRTUAL ADDRESS TO 0 MAPPED THRU PAR2
1330  004650  005037  172344            CLR     @#KIPAR2        ;INIT PAR2 TO LOC 0
1331  004654  012737  000200  172346    MOV     #200,@#KIPAR3   ;SET PAR3 TO 2ND 4K BANK
1332  004662  012737  004710  000004    MOV     #3$,@#ERRVEC    ;LOAD TIME OUT VECTOR
1333
1334  004670  012703  000001      1$:   MOV     #BIT0,R3        ;SET-UP 8K BANK POINTER
1335
1336  004674  011222              2$:   MOV     (R2),(R2)+      ;READ AND WRITE ALL MEMORY
1337  004676  032702  037777            BIT     #MASK8K,R2      ;IF NOT 8K BOUNDARY
1338  004702  001374                    BNE     2$              ;  THEN TRY SOMEMORE
1339  004704  050310                    BIS     R3,(R0)         ;  ELSE SET BANK FLAG IN MEMMAP
1340  004706  000424                    BR      5$              ;AND GO UPDATE VARIABLES AND CONTINUE
1341
1342                              ;*  TIMEOUT TRAPS TO HERE
1343  004710  062706  000004      3$:   ADD     #4,SP           ;RESTORE STACK POINTER
1344  004714  022702  060000            CMP     #60000,R2       ;IF NOT POSSIBLY THE I/O PAGE
1345  004720  001017                    BNE     5$              ;  THEN GO TEST SOMEMORE
1346  004722  005767  173610            TST     MMAVA           ;  ELSE IF 22 BIT ADDRESSING
1347  004726  100406                    BMI     4$              ;    THEN GO SEE IF 2M I/O PAGE
1348  004730  022737  007600  172346    CMP     #7600,@#KIPAR3  ;    ELSE IF NOT I/O BOUNDARY FOR 18 BITS
1349  004736  001010                    BNE     5$              ;      THEN GO UPDATE VARIABLES AND TRY SOMEMORE
1350  004740  050310                    BIS     R3,(R0)         ;      ELSE SET BANK EXISTS IN MEMMAP
1351  004742  000433                    BR      7$              ;AND GO TYPE MEMORY MAP
1352
1353  004744  022737  177600  172346  4$:   CMP     #177600,@#KIPAR3        ;IF NOT 2M I/O BOUNDARY
1354  004752  001002                    BNE     5$              ;  THEN GO TRY SOMEMORE SIZING
1355  004754  050310                    BIS     R3,(R0)         ;  ELSE SET BANK IN MEMMAP
1356  004756  000425                    BR      7$              ;AND GO TYPE MEMORY MAP
1357
1358  004760  062737  000400  172344  5$:   ADD     #400,@#KIPAR2   ;UPDATE MAP TO NEXT
1359  004766  062737  000400  172346        ADD     #400,@#KIPAR3   ;    8K BANK
1360  004774  012702  040000            MOV     #40000,R2       ;RESTORE ADDRESS POINTER TO 1ST ADDRESS OF THIS BANK
1361  005000  005767  173532            TST     MMAVA           ;IF NOT 18 BIT ADDR
1362  005004  100403                    BMI     6$              ;THEN GO TEST SOMEMORE
1363  005006  006303                    ASL     R3              ;18 BIT ADDR. = 1 WORD
1364  005010  001331                    BNE     2$              ;IF NOT END OF 18 BIT ADDR
1365                                                            ;  THEN GO SIZE SOMEMORE
1366  005012  000407                    BR      7$              ;  ELSE ALL DONE 18 BIT SIZING.
1367  005014  006303            6$:     ASL     R3              ;UPDATE BANK POINTER
1368  005016  001326                    BNE     2$              ;IF NOT END OF THIS MEMMAP ENTRY, THEN CONTINUE
1369  005020  062700  000002            ADD     #2,R0           ;  ELSE UPDATE TO NEX MEMMAP ENTRY
1370  005024  022700  001554            CMP     #MEMMAP+40,R0   ;IF NOT END OF MEMMAP TABLE
1371  005030  001317                    BNE     1$              ;    THEN GO SIZE SOMEMORE
1372                                                            ;    ELSE ALL DONE SIZING
1373  005032  000400            7$:     BR      DISMAP          ;GO TYPE OUT MEMORY MAP
1374
1375
1376                              ;;*****************************************************************
1377                              ;*  ROUTINE WILL TYPE OUT MEMMAP, LOAD TEST MAP (SAVTST) AND CHECKS
1378                              ;*  TO INSURE LOWEST 16K OF MEMORY IS AVAILABLE FOR TEST TO RUN
1379                              ;*      STORAGE LOCATIONS:
1380                              ;*              R0      = MEMMAP POINTER ... LO 128K
1381                              ;*              R1      = COUNTER
1382                              ;*              R2      = SAVTST POINTER ... LO 128K
1383                              ;;*****************************************************************
```

```
1384  005034  012700  001514          DISMAP: MOV     #MEMMAP,R0      ;LOAD R0 WITH MEMMAP ADR
1385  005040  012737  023252  000004          MOV     #ERRTRP,@#ERRVEC;SET UP TIME OUT VECTOR
1386  005046  004567  014522                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
1387  005052  023564                          .WORD   MEMMES          ;ADDRESS OF MESSAGE TO BE TYPED
1388                                                                  ;'MEMORY MAP:'
1389  005054  004767  011570                  JSR     PC,TYPMAP       ;GO TYPE THE MAP
1390  005060  004567  014510                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
1391  005064  001171                          .WORD   $CRLF           ;ADDRESS OF MESSAGE TO BE TYPED
1392  005066  012702  001614                  MOV     #SAVTST,R2      ;LOAD ADR OF SAVTST TABLE TO BE CLEARED
1393  005072  012700  001514                  MOV     #MEMMAP,R0      ;LOAD ADR OF MEMMAP TABLE
1394  005076  012701  000020                  MOV     #16.,R1         ;LOAD COUNTER
1395  005102  005012                  1$:     CLR     (R2)            ;CLEAR SAVTST TABLE ENTRY
1396  005104  012022                          MOV     (R0)+,(R2)+     ;LOAD SAVTST FROM MEMMAP
1397  005106  077103                          SOB     R1,1$           ;DECREMENT CTR 16 TIMES
1398  005110  016700  174400                  MOV     MEMMAP,R0       ;LOAD R0 WITH MAP OF 1ST 128K
1399  005114  042700  177774                  BIC     #177774,R0      ;MASK ALL BUT BOTTOM 16K
1400  005120  022700  000003                  CMP     #3,R0           ;IF BOTTOM 16K IS ALL THERE
1401  005124  001404                          BEQ     GMPR            ;   THEN GO RUN
1402  005126  004567  014442                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
1403  005132  023644                          .WORD   INSUFF          ;ADDRESS OF MESSAGE TO BE TYPED
1404                                                                  ;'FIRST 16K OF MEMORY NOT ALL THERE.'
1405  005134  000000                          HALT                    ;FATAL ERROR HALT ...
1406                                                                  ;MEMORY IS NOT CONFIGURED TO RUN THIS PROGRAM
1407
1408
1409                                  .SBTTL  MAP PARITY REGISTERS
1410                                  ;;*****************************************************************
1411                                  ;*  SEARCH FOR PARITY REGISTERS PRESENT AND TYPE ADDRESSES OF THOSE FOUND
1412                                  ;*  THAT ARE FUNCTIONAL AND HAVE CORRESPONDING PARITY MEMORY
1413                                  ;;*****************************************************************
1414
1415  005136  012704  003370          GMPR:   MOV     #MPRX,  R4      ;SET UP POINTER TO PARITY REG EXIST TABLE.
1416  005142  032777  000100  173760          BIT     #SW06,  @SWR    ;CHECK FOR INHIBIT PARITY SWITCH.
1417  005150  001040                          BNE     GMPRD           ;BR IF INHIBIT PARITY.
1418  005152  012703  002270                  MOV     #MPR0,  R3      ;SET UP TABLE POINTER
1419  005156  012737  005200  000004          MOV     #GMPRB, @#ERRVEC ;SET UP TIMEOUT TRAP SERVICE
1420  005164  042713  000001          GMPRA:  BIC     #1,     (R3)    ;CLEAR FLAG BIT IN TABLE
1421  005170  005773  000000                  TST     @(R3)           ;DOES THIS MEMORY PARITY REGISTER EXIST.
1422                                  ;*  IF IT DOESN'T EXIST, A TIMEOUT TRAP WILL GO TO 'GMPRB'.
1423  005174  012324                          MOV     (R3)+,  (R4)+   ;SAVE IT IN THE PARITY REG EXIST TABLE.
1424  005176  000404                          BR      GMPRBA          ;SKIP TIMEOUT SERVICE CODE
1425                                  ;*  TIMEOUT COMES HERE
1426  005200  062706  000004          GMPRB:  ADD     #4,SP           ;RESTORE STACK POINTER
1427  005204  052723  000001                  BIS     #1,     (R3)+   ;SET FLAG TO INDICATE REGISTER NOT PRESENT
1428  005210  012701  000021          GMPRBA: MOV     #17.,R1         ;LOAD COUNTER
1429  005214  005023                  GMPRC:  CLR     (R3)+           ;CLEAR ENTRY IN THIS GPR TABLE
1430  005216  077102                          SOB     R1,GMPRC        ;DECREMENT COUNTER AND CONTINUE
1431                                                                  ;CLEARING COUNTER UNTIL COMPLETED.
1432  005220  020327  003370                  CMP     R3,     #MPRX   ;HAVE WE CHECKED ALL REGISTERS?
1433
1434  005224  103757                          BLO     GMPRA           ;NO - GO BACK TO CHECK NEXT ONE
1435  005226  005014                          CLR     (R4)            ;SET TERMINATOR IN PARITY REG EXIST TABLE.
1436  005230  012737  023252  000004          MOV     #ERRTRP,@#ERRVEC ; RESTORE TRAPCATCHER
1437  005236  005767  176126                  TST     MPRX                    ;ANY PARITY REGISTERS PRESENT?
1438  005242  001006                          BNE     MPAMEM                  ;YES - GO TEST CONTROLS PRESENT
1439  005244  004567  014324                  JSR     R5,     $PRINT  ;G' PRINT OUT THE FOLLOWING MESSAGE.
```

```
1440  005250  023725                      .WORD   MTR             ;ADDRESS OF MESSAGE TO BE TYPED
1441                                                              ;'NO MEMORY PARITY REGISTERS FOUND''
1442  005252  005014              GMPRD:  CLR     (R4)            ;MAKE SURE TABLE IS CLEAR.
1443  005254  000167  000744              JMP     MANUAL                  ;AND SKIP ALL CONTROLS TESTING
1444
```

```
                                .SBTTL  MAP PARITY MEMORY
                                ;;*****************************************************************
                                ;*  MAP CORRESPONDENCE BETWEEN PARITY REGISTERS AND MEMORY,
                                ;*  AND TYPE RESULTS.  SET WRITE WRONG PARITY IN ALL
                                ;*  REGISTERS PRESENT, THEN WRITE TEST LOCATION VIA DATO & READ TEST
                                ;*  LOCATION VIA DATI, THEN CLEAR WRITE WRONG PARITY IN ALL REGISTERS.
                                ;*      NOTE:  THAT IF PARITY MEMORY IS NOT LOCATED CORRECTLY THAT
                                ;*             IT IS IN ALL PROBABILITY DUE TO ONE OF THE FOLLOWING
                                ;*             FAILURES:
                                ;*              - SETTING WRITE WRONG PARITY DIDN'T CAUSE BAD PARITY TO BE WRITTEN
                                ;*              - PARITY GENERATE OR DETECT LOGIC FAILED
                                ;*              - PARITY ERROR BIT FAILED TO SET
                                ;*              - PARITY BITS IN MEMORY LOCATION FAILED
                                ;*              - I.E. BIT STUCK AT GOOD PARITY VALUE
                                ;*      STORAGE USED:
                                ;*             R0      = MEMMAP & PMEMAP TABLE INDEX
                                ;*             R1      = BANK POINTER
                                ;*             R2      = ADDRESS TO WRITE WRONG PARITY TO
                                ;*             R3      = MPR TABLE POINTER
                                ;*             R4      = ADDRESS OF PRESENT MPR TABLE ENTRY
                                ;*             R5      = MPR TABLES INDEX
                                ;;*****************************************************************
005260  004767  011230        MPAMEM: JSR     PC,CLRPAR       ;INITIALIZE ALL PARITY REGISTERS
005264  012702  014000                MOV     #14000,R2       ;SET ADDRESS TO 14000 TO WRITE WRONG PARITY
005270  005767  173242                TST     MMAVA           ;IF NO MEM MGMT
005274  001404                        BEQ     MAPRB           ;  THEN GO MAP PARITY MEMORY
005276  012702  054000                MOV     #54000,R2       ;  ELSE SET ADDRESS POINTER TO MAP THRU PAR2
005302  004767  005330                JSR     PC,MMINIT       ;SET UP MEM MGMT REGISTERS

005306  012703  001654        MAPRB:  MOV     #PMEMAP,R3      ;LOAD PMEMAP TABLE ADR
005312  012704  000020                MOV     #16.,R4         ;LOAD COUNTER
005316  005023                1$:     CLR     (R3)+           ;CLEAR ALL OF TABLE
005320  077402                        SOB     R4,1$           ;IF NOT DONE CLEARING THEN TRY SOMEMORE

005322  004767  011166                JSR     PC,CLRPAR       ;GO INITIALIZE ALL PARITY REGISTERS
005326  005000                        CLR     R0              ;INIT INDEX FOR MEMMAP AND PMEMAP TABLES
005330  012705  000002                MOV     #2,R5           ;INIT INDEX FOR MPR TABLES
005334  012701  000001                MOV     #BIT0,R1        ;INIT BANK POINTER

005340  012703  002270        2$:     MOV     #MPR0,R3        ;INIT MPR  TABLE ADDRESS POINTER
005344  010304                3$:     MOV     R3,R4           ;UPDATE TO NEW TABLE
005346  060504                        ADD     R5,R4           ;UPDATE INDEX THRU NEW TABLE

005350  032713  000001                BIT     #BIT0,(R3)      ;IF CSR IS NOT PRESENT
005354  001021                        BNE     4$              ;  THEN GO TRY AGAIN
005356  013773  001766  000000        MOV     @#WWP,@(R3)     ;  ELSE SET WRITE WRONG PARITY BIT
005364  011212                        MOV     (R2),(R2)       ;WRITE WRONG PARITY
005366  005712                        TST     (R2)            ;READ WRONG PARITY
005370  043773  001766  000000        BIC     @#WWP,@(R3)     ;CLEAR WRITE WRONG PARITY BIT
005376  005773  000000                TST     @(R3)           ;IF NO PARITY ERROR
005402  100006                        BPL     4$              ;  THEN REGISTER DOES NOT CONTROL THIS MEMORY
005404  012763  070032  000042        MOV     #70032,42(R3)   ;  ELSE SAVE THE PARITY MASK IN TABLE
005412  050160  001654                BIS     R1,PMEMAP(R0)   ;SET BANK IN PMEMAP
005416  050114                        BIS     R1,(R4)         ;SET BANK IN MPR TABLES
005420  062703  000044        4$:     ADD     #44,R3          ;UPDATE TO  NEXT MPR TABLE
005424  022703  003370                CMP     #MPRX,R3        ; IF NOT END OF TABLE
```

```
1501  005430  101345                          BHI     3$              ;  THEN TRY SOMEMORE
1502  005432  011212                          MOV     (R2),(R2)       ;  ELSE CLEAR BAD PARITY
1503
1504  005434  005767  173076          5$:     TST     MMAVA           ;IF NO MEM MGMT
1505  005440  001417                          BEQ     6$              ;  THEN GO UPDATE FOR 16 BIT SYSTEM
1506  005442  062737  000400  172344          ADD     #400,@#KIPAR2   ;  ELSE UPDATE PAR2 TO NEXT 8K BANK
1507  005450  006301                          ASL     R1              ;UPDATE BANK POINTER TO NEXT BANK
1508  005452  001020                          BNE     7$              ;IF STILL SOME TO CHECK IN THIS 128K BANK THEN DO IT
1509  005454  062700  000002                  ADD     #2,R0           ;  ELSE UPDATE INDEX FOR MEMMAP AND PMEMAP TABLES
1510  005460  062705  000002                  ADD     #2,R5           ;UPDATE INDEX FOR GPR TABLES
1511  005464  012701  000001                  MOV     #BIT0,R1        ;INIT BANK POINTER
1512  005470  022700  000040                  CMP     #40,R0          ;IF END OF MEMMAP
1513  005474  001413                          BEQ     9$              ; THEN GO TYPE MEM PARITY MAPS
1514  005476  000406                          BR      7$              ;GO TRY SOMEMORE
1515
1516  005500  062702  040000          6$:     ADD     #40000,R2       ;UPDATE ADDRESS TO NEXT 8K BANK
1517  005504  106301                          ASLB    R1              ;UPDATE BANK POINTR
1518  005506  032701  000020                  BIT     #BIT4,R1        ;IF DONE WITH 16 BITS
1519  005512  001004                          BNE     9$              ;  THEN FINISHED
1520
1521  005514  036001  001514          7$:     BIT     MEMMAP(R0),R1   ;IF BANK DOES NOT EXIST
1522  005520  001745                          BEQ     5$              ;  THEN GET ANOTHER BANK
1523
1524  005522  000706                          BR      2$              ;  THEN GO DO SOMEMORE
1525
1526  005524  000167  000000          9$:     JMP     TMAP            ;GO TYPE PARITY MAPS
1527
1528                                  .SBTTL  DISPLAY PARITY MEMORY MAP
1529                                  ;;*****************************************************************
1530                                  ;*  ROUTINE TO TYPE MAP OF WHERE PARITY MEMORY IS PRESENT AND WHICH
1531                                  ;*  CONTROL REGISTERS CONTROL WHICH MEMORY.
1532                                  ;*      STORAGE USED:
1533                                  ;*              R0      = FIRST ADDRESS OF MAP TO BE TYPED
1534                                  ;*              R1      = PARITY REGISTER ADDRESS ... BITS 14-0
1535                                  ;*              R2      = PARITY REGISTER ADDRESS ... BITS 21-15
1536                                  ;*              R3      = MPR TABLE ENTRY
1537                                  ;;*****************************************************************
1538  005530  004767  010760          TMAP:   JSR     PC,CLRPAR       ;INITIALIZE ALL PARITY REGISTERS PRESENT
1539  005534  004567  014034                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
1540  005540  023602                          .WORD   MTMAP           ;ADDRESS OF MESSAGE TO BE TYPED
1541                                                                  ;'PARITY MEMORY MAP:''
1542  005542  012703  002270                  MOV     #MPR0,R3        ;INIT MPR TABLE POINTER
1543
1544  005546  032713  000001          1$:     BIT     #BIT0,(R3)      ;IF THIS REGISTER IS NOT PRESENT
1545  005552  001050                          BNE     6$              ;  THEN GO TRY AGAIN
1546
1547  005554                          2$:
1548  005554  004567  014014                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
1549  005560  024211                          .WORD   MX1             ;ADDRESS OF MESSAGE TO BE TYPED
1550                                                                  ;'PARITY REGISTER AT''
1551  005562  011301                          MOV     (R3),R1         ;SAVE PARITY REGISTER ADDRESS
1552  005564  042701  100000                  BIC     #100000,R1      ;DEVELOP BITS 14-0 OF REGISTER ADDRESS
1553  005570  005767  172742                  TST     MMAVA           ;IF TYPE OF MEMORY MANAGEMENT
1554  005574  100404                          BMI     3$              ;  THEN 22 BIT BRANCH
1555  005576  001006                          BNE     4$              ;  OR 18 BIT BRANCH
1556  005600  012702  000001                  MOV     #1,R2           ;  ELSE 16 BIT SET BITS 21-15 FOR PRINT OUT
```

```
1557  005604  000405                          BR      5$              ;AND GO TYPE OUT
1558
1559  005606  012702  000177        3$:       MOV     #177,R2         ;LOAD BITS 21-15 FOR PRINT OUT
1560  005612  000402                          BR      5$              ;AND GO TYPE OUT
1561
1562  005614  012702  000007        4$:       MOV     #7,R2           ;LOAD BITS 21-15 FOR PRINT OUT
1563
1564  005620                        5$:
1565  005620  010246                          MOV     R2,-(SP)        ;;SAVE R2 FOR TYPEOUT
1566                                                                  ;;TYPE ADDRESS BITS 21-15
1567                                ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOS ROUTINE
1568                                ;*  WIHTOUT USING A ''TRAP'' INSTRUCTION AS CALLED FOR BY **SYSMAC**.
1569  005622  106746                          MFPS    -(SP)           ;PUT THE PROCESSOR STATUS ON THE STACK
1570  005624  105066  000001                CLRB    1(SP)           ;HIGH BYTE CLEARED TO INSURE KERNEL MODE
1571                                                                  ;ON PSW RETURN.
1572  005630  004767  015166                JSR     PC,     $TYPOS  ;GO TO THE SUBROUTINE
1573  005634     003                        .BYTE   3               ;;TYPE 3 DIGIT(S)
1574  005635     000                        .BYTE   0               ;;SUPPRESS LEADING ZEROS
1575  005636  010146                        MOV     R1,-(SP)        ;;SAVE R1 FOR TYPEOUT
1576                                                                  ;;TYPE ADDRESS BITS 14-0
1577                                ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOS ROUTINE
1578                                ;*  WIHTOUT USING A ''TRAP'' INSTRUCTION AS CALLED FOR BY **SYSMAC**.
1579  005640  106746                          MFPS    -(SP)           ;PUT THE PROCESSOR STATUS ON THE STACK
1580  005642  105066  000001                CLRB    1(SP)           ;HIGH BYTE CLEARED TO INSURE KERNEL MODE
1581                                                                  ;ON PSW RETURN.
1582  005646  004767  015150                JSR     PC,     $TYPOS  ;GO TO THE SUBROUTINE
1583  005652     005                        .BYTE   5               ;;TYPE 5 DIGIT(S)
1584  005653     001                        .BYTE   1               ;;TYPE LEADING ZEROS
1585  005654  004567  013714                JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
1586  005660  024237                        .WORD   MX2             ;ADDRESS OF MESSAGE TO BE TYPED
1587                                                                  ;''CONTROLS''
1588  005662  010300                        MOV     R3,R0           ;SET UP R0 FOR TYPMAP ROUTINE
1589  005664  062700  000002                ADD     #2,R0           ;POINT TO MAP ENTRY FOR MPR TABLE
1590  005670  004767  010754                JSR     PC,TYPMAP       ;GO TYPE MEMORY COVERED BY THIS REGISTER
1591
1592  005674  062703  000044        6$:       ADD     #44,R3          ;UPDATE TO NEXT REGISTER IN TABLE
1593  005700  022703  003370                  CMP     #MPRX,R3        ;IF NOT END OF MPR TABLE
1594  005704  101320                          BHI     1$              ;  THEN DO SOMEMORE
1595
1596  005706  012700  000010                MOV     #10,R0          ;LOAD DELAY
1597  005712  012701  177777        7$:       MOV     #-1,R1          ;LOAD DELAY
1598  005716  077101                  8$:       SOB     R1,8$           ;ALLOW DELAY TO INSURE PRINT OUT IS
1599  005720  077004                          SOB     R0,7$           ;COMPLETED BEFORE RESET OCCURS
1600
1601  005722  005737  003370                TST     @#MPRX          ;IF PARITY REGISTERS TO TEST
1602  005726  001002                        BNE     CTRLS           ;  THEN GO TEST
1603  005730  000167  000270                JMP     MANUAL          ;  ELSE JUMP OVER TESTS
1604
1605
1606
1607                                .SBTTL  TEST PARITY REGISTERS
1608                                ;;*******************************************************************
1609                                ;* SHOW THAT BITS 0, 2, 5 - 11 AND 15 OF EACH PARITY REGISTER PRESENT
1610                                ;*  CAN BE SET AND CLEARED.
1611                                ;*  THIS IS A ONCE ONLY TEST.
1612                                ;;*******************************************************************
```

```
1613
1614  005734  012703  002270          CTRLS:  MOV     #MPR0,  R3      ;LOAD INITIAL TABLE ADDRESS FOR A POINTER
1615  005740  011302                  1$:     MOV     (R3),R2         ;GET CSR ADDRESS INTO R2
1616  005742  032702  000001                  BIT     #BIT0,R2        ;  ELSE IF CSR DOES NOT EXIST
1617  005746  001052                          BNE     105$            ;    THEN BRANCH
1618  005750  016367  000042  173530          MOV     42(R3),RESRVD   ;    ELSE LOAD MASK, IF EQ TO 0
1619  005756  001446                          BEQ     105$            ;      THEN DO NOT TEST
1620  005760  012700  000001                  MOV     #1,R0           ;LOAD R0 WITH 1ST BIT TO BE TESTED
1621  005764  005012                          CLR     (R2)            ;INITIALIZE THE PARITY REGISTER
1622  005766  011201                          MOV     (R2),R1         ;READ THE CONTENTS OF THE PARITY REGISTER
1623  005770  046701  173512                  BIC     RESRVD,R1       ;CLEAR RESERVED BITS, IF EQ 0
1624  005774  001405                          BEQ     2$              ;  THEN BRANCH
1625  005776  004767  010534          64$:    JSR     PC,     SPRNT   ;SET UP VALUES FOR ERROR PRINTING.
1626  006002  004767  011660                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
1627  006006  000001                          .WORD   1               ;ERROR TYPE CODE.
1628  006010  030067  173472          2$:     BIT     R0,RESRVD       ;IF THIS BIT IS RESERVED
1629  006014  001025                          BNE     3$              ;  THEN BRANCH AND DON'T TEST
1630  006016  010012                          MOV     R0,(R2)         ;  ELSE SET THIS BIT IN CSR
1631  006020  011201                          MOV     (R2),R1         ;READ AND SAVE CONTENTS OF CSR
1632  006022  005012                          CLR     (R2)            ;CLEAR THE CSR
1633  006024  046701  173456          106$:   BIC     RESRVD,R1       ;CLEAR RESERVED BITS
1634  006030  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
1635  006032  001405                          BEQ     66$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
1636  006034  004767  010540          65$:    JSR     PC,     SPRNT0  ;SET UP VALUES FOR ERROR PRINTING.
1637  006040  004767  011622                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
1638  006044  000001                          .WORD   1               ;ERROR TYPE CODE.
1639  006046                          66$:
1640                                                                  ;MAKE SURE BIT WAS CLEARED OUT OF CSR
1641  006046  011201                          MOV     (R2),R1         ;READ THE CONTENTS OF THE PARITY REGISTER
1642  006050  046701  173432                  BIC     RESRVD, R1      ;CLEAR BITS WHICH ARE RESERVED
1643  006054  001405                          BEQ     3$              ;CHECK OTHER BITS - BRANCH IF OK
1644  006056  004767  010454          67$:    JSR     PC,     SPRNT   ;SET UP VALUES FOR ERROR PRINTING.
1645  006062  004767  011600                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
1646  006066  000001                          .WORD   1               ;ERROR TYPE CODE.
1647  006070  006300                  3$:     ASL     R0              ;ROTATE TO GET NEXT BIT TO BE TESTED
1648  006072  001346                          BNE     2$              ;BRANCH IF NOT DONE WITH ALL BITS
1649  006074  062703  000044          105$:   ADD     #44,R3          ;UPDATE PTR TO NEXT ENTRY
1650  006100  022703  003370                  CMP     #MPRX,R3        ;IF NOT DONE WITH TABLE
1651  006104  003315                          BGT     1$              ;THEN TRY AGAIN
1652
1653                                  ;;*****************************************************************
1654                                  ;* SHOW THAT RESET CLEARS BITS 0,2 AND 15 OF EACH PARITY REGISTER PRESENT.
1655                                  ;* ALSO BIT 14 IN PARITY CSR IF MEMORY IS SET FOR 22 BIT ADDRESSING
1656                                  ;*  THIS IS A ONCE ONLY TEST.
1657                                  ;;*****************************************************************
1658
1659  006106  012704  002270          RESCHK: MOV     #MPR0,  R4      ;LOAD INITIAL TABLE ADDRESS FOR A POINTER
1660  006112  022704  003370          1$:     CMP     #MPRX,R4        ;IF END OF TABLE
1661  006116  003411                          BLE     100$            ;      THEN BRANCH
1662  006120  032714  000001                  BIT     #BIT0,(R4)      ;      ELSE IF CSR DOES NOT EXIS
1663  006124  001003                          BNE     101$            ;       THEN BRANCH
1664  006126  012774  177777  000000          MOV     #-1,@(R4)       ;       ELSE LOAD CSR WITH ALL 1'S
1665  006134  062704  000044          101$:   ADD     #44,R4          ;UPDATE POINTER
1666  006140  000764                          BR      1$              ;TRY AGAIN
1667  006142  000005                  100$:   RESET                   ;RESET THE WORLD
1668  006144  012702  002270                  MOV     #MPR0,R2        ;LOAD INITIAL ADDRESS FOR POINTER
```

```
1669  006150  022702  003370          2$:     CMP     #MPRX,R2        ;IF END OF TABLE
1670  006154  003423                          BLE     MANUAL          ;     THEN BRANCH
1671  006156  032712  000001                  BIT     #BIT0,(R2)      ;     ELSE IF CSR DOES NOT EXIST
1672  006162  001015                          BNE     65$             ;       THEN BRANCH
1673  006164  017201  000000                  MOV     @(R2),R1        ;       ELSE SAVE CONTENTS OF CSR
1674  006170  005072  000000                  CLR     @(R2)           ;CLEAR THE CSR
1675  006174  042701  037772                  BIC     #37772,R1       ;TEST FOR BIT14 TOO
1676                                                                  ;18 BIT MODE = BIT14 ALWAYS READ AS ZERO
1677                                                                  ;22 BIT MODE = BIT14 R/W,CLEARED BY RESET
1678  006200  005701                  4$:     TST     R1              ;CHECK IF REST  WERE CLEARED BY  RESET
1679  006202  001405                          BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
1680  006204  004767  010326          64$:    JSR     PC,     $PRNT   ;SET UP VALUES FOR ERROR PRINTING.
1681  006210  004767  011452                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
1682  006214  000001                          .WORD   1               ;ERROR TYPE CODE.
1683  006216                          65$:
1684  006216  062702  000044                  ADD     #44,R2          ;UPDATE POINTER
1685  006222  000752                          BR      2$              ;BRANCH BACK TO CHECK NEXT REGISTER
1686
1687
1688  006224  005067  173532          MANUAL: CLR     FSTADR          ;INIT FIRST ADDRESS
1689  006230  105767  173522                  TSTB    SELFLG          ;CHECK FOR SELECT PARAMETER SETUP
1690  006234  001002                          BNE     MANUL1          ;IF FLAG SET GET USERS PARAMETERS
1691  006236  000167  000402                  JMP     MANUL2          ;ELSE USE DEFAULT DATA
1692  006242                          MANUL1:
1693  006242  004567  013326                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
1694  006246  024275                          .WORD   FADMES          ;ADDRESS OF MESSAGE TO BE TYPED
1695                                                                  ;'FIRST ADDRESS:'
1696                                                                  ;FIRST ADDRESS 8K BOUNDARY
1697                                  ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $RDOCT ROUTINE
1698                                  ;*  WIHTOUT USING A 'TRAP' INSTRUCTION AS CALLED FOR BY **SYSMAC**.
1699  006250  106746                          MFPS    -(SP)           ;PUT THE PROCESSOR STATUS ON THE STACK
1700  006252  105066  000001                  CLRB    1(SP)           ;HIGH BYTE CLEARED TO INSURE KERNEL MODE
1701                                                                  ;ON PSW RETURN.
1702  006256  004767  013136                  JSR     PC,     $RDOCT  ;GO TO THE SUBROUTINE
1703  006262  032716  037777                  BIT     #37777,(SP)     ;ON 8K BOUNDARY?
1704  006266  001365                          BNE     MANUL1          ;IF NO,REASK
1705  006270  016703  013276                  MOV     $HIOCT,R3       ;SAVE ORIGINAL HIGH BITS
1706  006274  011604                          MOV     (SP),R4         ;AND LOW BITS
1707  006276  006267  013270                  ASR     $HIOCT          ;DIVIDE HIGH ADDRESS INTO
1708  006302  006016                          ROR     (SP)            ;NUMBER OF 128K BANKS AND
1709  006304  006267  013262                  ASR     $HIOCT          ;NUMBER OF 8K BANK WITHIN
1710  006310  006016                          ROR     (SP)            ;128K BANK
1711  006312  006367  013254                  ASL     $HIOCT          ;MAKE NUMBER OF 128K BANKS INDEX
1712  006316  012700  001714                  MOV     #BITPT,R0       ;GET START OF TEST TABLE
1713  006322  066700  013244                  ADD     $HIOCT,R0       ;ADD INDEX
1714  006326  012701  000012                  MOV     #12,R1          ;SET UP TO ALIGN 8K BANK COUNT
1715  006332  006016                  1$:     ROR     (SP)            ;ALIGN COUNT
1716  006334  077102                          SOB     R1,1$           ;LOOP UNTIL DONE
1717  006336  052710  000001                  BIS     #BIT0,(R0)      ;INIT 8K BANK POINTER IN 128K WORD
1718  006342  005716                  2$:     TST     (SP)            ;SHIFT BIT UNTIL COUNT = 0
1719  006344  001403                          BEQ     3$              ;WHEN ZERO  DONE
1720  006346  006310                          ASL     (R0)            ;SHIFT BIT
1721  006350  005316                          DEC     (SP)            ;SUBTRACT FROM COUNT
1722  006352  000773                          BR      2$              ;LOOP BACK
1723  006354  012702  001514          3$:     MOV     #MEMMAP,R2      ;GET FIRST ADDRESS OF MEMORY TABLE
1724  006360  066702  013206                  ADD     $HIOCT,R2       ;ADD INDEX
```

```
1725  006364  031012                        BIT     (R0),(R2)       ;IS BIT JUST LOCATED,SET IN MEM. TABLE
1726  006366  001725                        BEQ     MANUL1          ;IF NO BANK EXISTS...REASK
1727  006370                          4$:
1728  006370  004567  013200                JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
1729  006374  024362                        .WORD   LADMES          ;ADDRESS OF MESSAGE TO BE TYPED
1730                                                                ;"LAST ADDRESS:"
1731                                  ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $RDOCT ROUTINE
1732                                  ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
1733  006376  106746                        MFPS    -(SP)           ;PUT THE PROCESSOR STATUS ON THE STACK
1734  006400  105066  000001                CLRB    1(SP)           ;HIGH BYTE CLEARED TO INSURE KERNEL MODE
1735                                                                ;ON PSW RETURN.
1736  006404  004767  013010                JSR     PC,     $RDOCT  ;GO TO THE SUBROUTINE
1737  006410  005716                        TST     (SP)            ;CHECK IF LOW BITS ZERO
1738  006412  001010                        BNE     5$              ;IF NOT,NO DEFAULT
1739  006414  005767  013152                TST     $HIOCT          ;CHECK IF HIGH BITS ZERO
1740  006420  001005                        BNE     5$              ;IF NOT,NO DEFAULT
1741  006422  016716  172526                MOV     $TMP2,(SP)      ;IF BOTH ZERO,FILL IN DEFAULT
1742  006426  016767  172524  013136        MOV     $TMP3,$HIOCT    ;FOR LOW AND HIGH BITS
1743  006434  020467  013132          5$:   CMP     R4,$HIOCT       ;CHECK FOR LAST ADDR. BELOW FIRST
1744  006440  101353                        BHI     4$              ;IF YES....REASK
1745  006442  103402                        BLO     6$              ;IF LAST HIGHER GO ON
1746  006444  020316                        CMP     R3,(SP)         ;IF EQUAL CHECK LOW BITS
1747  006446  101350                        BHI     4$              ;IF LOW BITS LOWER...REASK
1748  006450  012700  000020          6$:   MOV     #20,R0          ;TABLE COUNTER
1749  006454  012701  001714                MOV     #BITPT,R1
1750  006460  012702  001614                MOV     #SAVTST,R2
1751  006464  052122                  7$:   BIS     (R1)+,(R2)+     ;STORE BITPT IN SAVTST MAP
1752  006466  077002                        SOB     R0,7$           ;DO 16. TIMES
1753  006470  020367  013076                CMP     R3,$HIOCT       ;COMPARE HIGH BITS OF FIRST TO LAST
1754  006474  103403                        BLO     8$              ;IF LOWER SEE IF NEXT BANK EXISTS
1755  006476  101027                        BHI     11$             ;IF HIGHER MUST BE LAST BANK
1756  006500  020416                        CMP     R4,(SP)         ;IF EQUAL CHECK LOW BITS
1757  006502  101025                        BHI     11$             ;IF LOW BITS FIRST HIGHER LAST BANK
1758  006504  062704  040000          8$:   ADD     #40000,R4       ;UPDATE TO NEXT BANK
1759
1760  006510  005503                        ADC     R3
1761  006512  012700  000017                MOV     #17,R0          ;SET UP TO UPDATE POINTER
1762  006516  012701  001714                MOV     #BITPT,R1       ;GET START ADDRESS OF TABLE
1763  006522  006321                        ASL     (R1)+           ;ROTATE POINTER WITHIN
1764  006524  006121                  9$:   ROL     (R1)+           ;TABLE
1765  006526  077002                        SOB     R0,9$           ;UNTIL ALL LOCATIONS DONE
1766  006530  103423                        BCS     13$             ;BANK DOESN'T MAP
1767  006532  012700  000020                MOV     #20,R0          ;SET UP TO SEE IF ONE EXISTS
1768  006536  012701  001714                MOV     #BITPT,R1       ;GET START ADDRESS OF POINTER TABLE
1769  006542  012702  001514                MOV     #MEMMAP,R2      ;GET START ADDRESS OF MEMORY TABLE
1770  006546  032122                  10$:  BIT     (R1)+,(R2)+     ;TEST IF BANK EXISTS
1771  006550  001337                        BNE     6$              ;IF MATCH UPDATE
1772  006552  077003                        SOB     R0,10$          ;DO ALL OF TABLE
1773  006554  000753                        BR      8$
1774  006556  012700  000020          11$:  MOV     #20,R0          ;MAKE SURE LAST BANK
1775  006562  012701  001714                MOV     #BITPT,R1       ;WAS MAPPED BY MEMORY
1776  006566  012702  001514                MOV     #MEMMAP,R2      ;SIZING ROUTINE
1777  006572  032122                  12$:  BIT     (R1)+,(R2)+     ;
1778  006574  001007                        BNE     14$
1779  006576  077003                        SOB     R0,12$
1780  006600  012706  001100          13$:  MOV     #STACK,SP       ;RESET STACK
```

```
1781  006604  004567  012764                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
1782  006610  024405                          .WORD   BADADR          ;ADDRESS OF MESSAGE TO BE TYPED
1783                                                                  ;''?ADDRESS IN UNMAPPED BANK?''
1784  006612  000604                          BR      MANUAL          ;LOOP BACK AND START OVER
1785  006614  012706  001100          14$:    MOV     #STACK,SP       ;RESET STACK
1786  006620  004567  012750                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
1787  006624  024442                          .WORD   CONST           ;ADDRESS OF MESSAGE TO BE TYPED
1788                                                                  ;''SELECT CONSTANT: ''
1789                                  ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $RDOCT ROUTINE
1790                                  ;*  WIHTOUT USING A ''TRAP'' INSTRUCTION AS CALLED FOR BY **SYSMAC**.
1791  006626  106746                          MFPS    -(SP)           ;PUT THE PROCESSOR STATUS ON THE STACK
1792  006630  105066  000001                CLRB    1(SP)           ;HIGH BYTE CLEARED TO INSURE KERNEL MODE
1793                                                                  ;ON PSW RETURN.
1794  006634  004767  012560                  JSR     PC,     $RDOCT  ;GO TO THE SUBROUTINE
1795  006640  012667  173120                  MOV     (SP)+,.CONST    ;SAVE THE DATA
1796  006644  032777  000040  172256  MANUL2: BIT     #SW05,@SWR      ;IS VECTOR AREA PROTECTED?
1797  006652  001403                          BEQ     START1          ;IF NO,GO ON
1798  006654  012767  001000  173100          MOV     #1000,FSTADR    ;IF YES,SET STARTING ADDRESS   1000
1799
1800

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

1807  006662  016706  173114          START1: MOV     .STACK, SP      ;SET STACK POINTER
1808  006666  012700  000020                  MOV     #20,R0          ;SET UP TO LOAD 20 LOCATIONS
1809  006672  016701  173110                  MOV     .SAVTST,R1      ;SOURCE TO BE SAVTST TABLE
1810  006676  016702  173106                  MOV     .TSTMAP,R2      ;DESTINATIION TO BE TEST MAP
1811  006702  012122                  1$:     MOV     (R1)+,(R2)+     ;INIT TESTING TABLE
1812  006704  077002                          SOB     R0,1$           ;LOOP UNTIL DONE
1813  006706  046767  171622  172640          BIC     PRGMAP,TSTMAP   ;DON'T TEST WHERE PROGRAM IS LOCATED.
1814  006714  012767  006662  172154          MOV     #START1,$LPADR  ;INIT LOOP ADDRESS.
1815  006722  066767  171604  172146          ADD     RELOCF, $LPADR
1816  006730  004767  006626                  JSR     PC,     MAMF    ;SET UP MEMORY PARITY ERROR VECTOR
1817  006734  005767  171576                  TST     MMAVA           ;CHECK FOR MEMORY MANAGEMENT AVAILABLE.
1818  006740  001406                          BEQ     TST1            ;BRANCH IF NO MEM MGMT.
1819  006742  032737  000001  177572          BIT     #BIT0,  @#SR0   ;CHECK IF MEM MGMT ENABLED.
1820  006750  001002                          BNE     TST1            ;BR IF MEM MGMT ENABLED.
1821  006752  004767  003660                  JSR     PC,     MMINIT  ;SET UP MEM MGMT REGISTERS.
1822
1823                                  .SBTTL SECTION 1:       MEMORY ADDRESS TESTS
1824                                  ;;*****************************************************************
1825                                  ;*TEST 1         WRITE VALUE OF MEMORY ADDRESS INTO MEMORY
1826                                  ;*       R0 = DATA WRITTEN INTO MEMORY (SHOULD BE)
1827                                  ;*       R1 = DATA READ FROM MEMORY (WAS)
1828                                  ;*       R2 = VIRTUAL ADDRESS
1829                                  ;*       R3 - NOT USED
1830                                  ;*       R4 = NOT USED
1831                                  ;*       R5 = BLOCK BOUNDARY BIT MASK.
1832                                  ;;*****************************************************************
1833  006756                          TST1:
1834  006756  004567  010372                  JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
1835                                  ;*  UPWARDS WORD ADDRESSING.
1836  006762  004467  004012                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
```

```
1837  006766  010200                  1$:     MOV     R2,     R0      ;SET UP TO CALCULATE PHYSICAL ADDRESS
1838  006770  004767  005124                  JSR     PC,     PHYADR  ;GET PHYSICAL ADDRESS INTO R0
1839  006774  010012                  2$:     MOV     R0,     (R2)    ;WRITE VALUE OF ADDRESS INTO ADDRESS
1840  006776  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
1841  007000  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
1842  007002  001405                          BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
1843  007004  004767  007614          64$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
1844  007010  004767  010652                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
1845  007014  000002                          .WORD   2               ;ERROR TYPE CODE.
1846  007016                          65$:
1847  007016  062700  000002                  ADD     #2,     R0      ;ADD #2 TO PHYSICAL ADDRESS
1848  007022  077514                          SOB     R5,     2$      ;BRANCH IF MORE IN CURRENT BLOCK.
1849  007024  004767  004162                  JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
1850
1851                                  ;*  CHECK THAT VALUE OF MEMORY ADDRESS WAS WRITTEN CORRECTLY
1852                                  ;*  DOWNWARDS WORD ADDRESSING.
1853  007030  004467  004042                  JSR     R4,     INITDN  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1854  007034  010200                  3$:     MOV     R2,     R0      ;SET UP TO CALCULATE PHYSICAL ADDRESS
1855  007036  162700  000002                  SUB     #2,     R0      ;GET LAST ADDRESS OF BLOCK
1856  007042  004767  005052                  JSR     PC,     PHYADR  ;GET PHYSICAL ADDRESS INTO R0
1857  007046  014201                  4$:     MOV     -(R2),  R1      ;GET THE DATA FROM MEMORY
1858  007050  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
1859  007052  001405                          BEQ     67$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
1860  007054  004767  007520          66$:    JSR     PC,     SPRNT0  ;SET UP VALUES FOR ERROR PRINTING.
1861  007060  004767  010602                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
1862  007064  000002                          .WORD   2               ;ERROR TYPE CODE.
1863  007066                          67$:
1864  007066  162700  000002                  SUB     #2,     R0      ;DEC DATA BY 2
1865  007072  077513                          SOB     R5,     4$      ;BRANCH IF MORE IN CURRENT BLOCK.
1866  007074  004767  004456                  JSR     PC,     MMDOWN  ;FIND NEXT BLOCK AND LOOP TO 3$.
```

```
                                ;;*****************************************************************
                                ;*TEST 2        WRITE VALUE OF MEMORY ADDRESS INTO MEMORY
                                ;*      R0 = DATA WRITTEN INTO MEMORY (SHOULD BE)
                                ;*      R1 = DATA READ FROM MEMORY (WAS)
                                ;*      R2 = VIRTUAL ADDRESS
                                ;*      R3 = NOT USED
                                ;*      R4 = NOT USED
                                ;*      R5 = BLOCK BOUNDARY BIT MASK.
                                ;;*****************************************************************
007100                          TST2:
007100  004567  010250                  JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
                                ;*  UPWARDS BYTE ADDRESSING.
007104  004467  003670                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
007110  010200                  1$:     MOV     R2,     R0      ;SET UP TO CALCULATE PHYSICAL ADDRESS
007112  004767  005002                  JSR     PC,     PHYADR  ;GET PHYSICAL ADDRESS INTO R0
007116  006305                          ASL     R5              ;MAKE TEST COUNTER BYTE VALUE
007120  110022                  2$:     MOVB    R0,     (R2)+   ;WRITE VALUE OF ADDRESS INTO ADDRESS
007122  005200                          INC     R0              ;ADD ONE TO PHYSICAL ADDRESS
007124  077503                          SOB     R5,     2$      ;BRANCH IF MORE IN CURRENT BLOCK.
007126  004767  004060                  JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.

                                ;*  CHECK THAT VALUE OF MEMORY ADDRESS WAS WRITTEN CORRECTLY
                                ;*  DOWNWARDS BYTE ADDRESSING.
007132  004467  003740                  JSR     R4,     INITDN  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
007136  010200                  3$:     MOV     R2,     R0      ;SET UP TO CALCULATE PHYSICAL ADDRESS
007140  005300                          DEC     R0              ;GET LAST BYTE ADDRESS OF BLOCK
007142  004767  004752                  JSR     PC,     PHYADR  ;GET PHYSICAL ADDRESS INTO R0
007146  006305                          ASL     R5              ;MAKE TEST COUNTER BYTE VALUE
007150  114201                  4$:     MOVB    -(R2),  R1      ;GET THE DATA FROM MEMORY
007152  120001                          CMPB    R0,     R1      ;CHECK THE DATA...LO BYTE ONLY VALID.
007154  001405                          BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
007156  004767  007416          64$:    JSR     PC,     SPRNT0  ;SET UP VALUES FOR ERROR PRINTING.
007162  004767  010500                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
007166  000003                          .WORD   3               ;ERROR TYPE CODE.
007170                          65$:
007170  005300                          DEC     R0              ;DEC DATA BY 1
007172  077512                          SOB     R5,     4$      ;BRANCH IF MORE IN CURRENT BLOCK.
007174  004767  004356                  JSR     PC,     MMDOWN  ;FIND NEXT BLOCK AND LOOP TO 3$.

                                ;;*****************************************************************
                                ;*TEST 3        WRITE 1'S COMPLEMENT VALUE OF ADDRESS INTO ADDRESS.
                                ;*      R0 = DATA WRITTEN INTO MEMORY (SHOULD BE)
                                ;*      R1 = DATA READ FROM MEMORY (WAS)
                                ;*      R2 = VIRTUAL ADDRESS
                                ;*      R3 = NOT USED
                                ;*      R4 = NOT USED
                                ;*      R5 = BLOCK BOUNDARY BIT MASK.
                                ;;*****************************************************************
007200                          TST3:
007200  004567  010150                  JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
                                ;*  DOWNWARDS WORD ADDRESSING.
007204  004467  003666                  JSR     R4,     INITDN  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
007210  010200                  1$:     MOV     R2,     R0      ;SET UP TO CALCULATE PHYSICAL ADDRESS
007212  162700  000002                  SUB     #2,     R0      ;GET LAST ADDRESS OF BLOCK
007216  004767  004676                  JSR     PC,     PHYADR  ;GET PHYSICAL ADDRESS INTO R0
007222  005100                          COM     R0              ;COMPLEMENT THE ADR
```

```
1923  007224  010042            2$:     MOV     R0,     -(R2)   ;PUT DATA INTO MEMORY
1924  007226  062700  000002            ADD     #2,     R0      ;+2 TO DATA--ADR GOES DOWN SO COM GOES UP
1925  007232  077504                    SOB     R5,     2$      ;BRANCH IF MORE IN CURRENT BLOCK.
1926  007234  004767  004316            JSR     PC,     MMDOWN  ;FIND NEXT BLOCK AND LOOP TO 1$.
1927
1928                              ;*  CHECK COMPLEMENT DATA WRITTEN DOWN
1929                              ;*  UPWARDS WORD ADDRESSING.
1930  007240  004467  003534            JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1931  007244  010200            3$:     MOV     R2,     R0      ;SET UP TO CALCULATE PHYSICAL ADDRESS
1932  007246  004767  004646            JSR     PC,     PHYADR  ;GET PHYSICAL ADDRESS INTO R0
1933  007252  005100                    COM     R0              ;COMPLEMENT IT
1934  007254                    4$:
1935  007254  012201                    MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
1936  007256  020001                    CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
1937  007260  001405                    BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
1938  007262  004767  007336  64$:     JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
1939  007266  004767  010374            JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
1940  007272  000002                    .WORD   2               ;ERROR TYPE CODE.
1941  007274                    65$:
1942  007274  162700  000002            SUB     #2,     R0      ;COUNT DOWN WITH ADDRESS
1943  007300  077513                    SOB     R5,     4$      ;BRANCH IF MORE IN CURRENT BLOCK
1944  007302  004767  003704            JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 3$.
1945
1946                              ;;*****************************************************************
1947                              ;*TEST 4         WRITE BANK # INTO ALL ADDRESSES IN A 8K BANK
1948                              ;*      R0 - DATA WRITTEN INTO MEMORY (SHOULD BE)
1949                              ;*      R1 = DATA READ FROM MEMORY (WAS)
1950                              ;*      R2 = VIRTUAL ADDRESS
1951                              ;*      R3 = NOT USED
1952                              ;*      R4 = NOT USED
1953                              ;*      R5 = BLOCK BOUNDARY BIT MASK.
1954                              ;;*****************************************************************
1955  007306                      TST4:
1956  007306  004567  010042            JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
1957                              ;*  UPWARDS BYTE ADDRESSING.
1958  007312  004467  003462            JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1959  007316  004767  004674    1$:     JSR     PC,     BANKNO  ;GET THE BANK NUMBER INTO R0
1960  007322  006305                    ASL     R5              ;MAKE TEST COUNTER BYTE VALUE
1961  007324  110022            2$:     MOVB    R0,     (R2)+   ;WRITE BANK # INTO ALL ADDRESSES
1962  007326  077502                    SOB     R5,     2$      ;BRANCH IF MORE IN CURRENT BLOCK.
1963  007330  004767  003656            JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
1964
1965                              ;*  CHECK THAT DATA WRITTEN ABOVE CAN BE READ
1966                              ;*  UPWARDS BYTE ADDRESSING.
1967  007334  004467  003440            JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1968  007340  004767  004652    3$:     JSR     PC,     BANKNO  ;GET THE BANK NUMBER INTO R0
1969  007344  006305                    ASL     R5              ;MAKE TEST COUNTER BYTE VALUE
1970  007346  112201            4$:     MOVB    (R2)+,  R1      ;READ THE DATA OUT OF MEMORY
1971  007350  120001                    CMPB    R0,     R1      ;CHECK DATA....LOW BYTE ONLY VALID
1972  007352  001405                    BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
1973  007354  004767  007226  64$:     JSR     PC,     SPRNT1  ;SET UP VALUES FOR ERROR PRINTING.
1974  007360  004767  010302            JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
1975  007364  000003                    .WORD   3               ;ERROR TYPE CODE.
1976  007366                    65$:
1977  007366  077511                    SOB     R5,     4$      ;BRANCH IF MORE IN CURRENT BLOCK
1978  007370  004767  003616            JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 3$.
```

```
                                ;;*****************************************************************
                                ;*TEST 5        WRITE 1'S COMPLEMENT OF BANK #.
                                ;*      R0 = DATA WRITTEN INTO MEMORY (SHOULD BE)
                                ;*      R1 = DATA READ FROM MEMORY (WAS)
                                ;*      R2 = VIRTUAL ADDRESS
                                ;*      R3 = NOT USED
                                ;*      R4 = NOT USED
                                ;*      R5 = BLOCK BOUNDARY BIT MASK.
                                ;;*****************************************************************
007374                          TST5:
007374  004567  007754                  JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
                                ;*  DOWNWARDS BYTE ADDRESSING.
007400  004467  003472                  JSR     R4,     INITDN  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
007404  004767  004606          1$:     JSR     PC,     BANKNO  ;GET THE BANK NUMBER INTO R0
007410  006305                          ASL     R5              ;MAKE TEST COUNTER BYTE VALUE
007412  005100                          COM     R0              ;1'S COMPLEMENT OF BANK #
007414  110042                  2$:     MOVB    R0,     -(R2)   ;PUT 1'S COM OF BANK # INTO MEMORY
007416  077502                          SOB     R5,     2$      ;BRANCH IF MORE IN CURRENT BLOCK.
007420  004767  004132                  JSR     PC,     MMDOWN  ;FIND NEXT BLOCK AND LOOP TO 1$.

                                ;*  CHECK THAT DATA WRITTEN CAN BE READ.
                                ;*  DOWNWARDS BYTE ADDRESSING.
007424  004467  003446                  JSR     R4,     INITDN  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
007430  004767  004562          3$:     JSR     PC,     BANKNO  ;GET THE BANK # INTO R0
007434  006305                          ASL     R5              ;MAKE TEST COUNTER BYTE VALUE
007436  005100                          COM     R0              ;SET 1'S COMPLEMENT OF BANK #
007440  114201                  4$:     MOVB    -(R2),  R1      ;READ DATA OUT OF MEMORY
007442  120001                          CMPB    R0,     R1      ;CHECK DATA...LOW BYTE ONLY VALID
007444  001405                          BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
007446  004767  007126          64$:    JSR     PC,     SPRNTO  ;SET UP VALUES FOR ERROR PRINTING.
007452  004767  010210                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
007456  000003                          .WORD   3               ;ERROR TYPE CODE.
007460                          65$:
007460  077511                          SOB     R5,     4$      ;BRANCH IF MORE IN CURRENT BLOCK.
007462  004767  004070                  JSR     PC,     MMDOWN  ;FIND NEXT BLOCK AND LOOP TO 3$.


                                ;*  IF PROGRAM HAS RELOCATED TO UPPER BOUNDARY,THE ADDRESSING
                                ;*  TESTS WILL BE EXECUTED FOR ALL BANKS AND WORST CASE NOISE TESTING
                                ;*  WILL BE LIMITED TO BANKS 0,1.
                                ;*  ALL OTHER BANKS WILL BE EXERCISED FOR WORST CASE NOISE TESTING
                                ;*  WHEN THE PROGRAM OCCUPIES BANKS 0,1.

007466  032767  000001  171040  SECT2:  BIT     #BIT0,PRGMAP    ;IS PROGRAM RELOCATED
007474  001014                          BNE     2$              ;IF NO,CONTINUE TESTING ALL BANKS
007476  012700  000020                  MOV     #20,R0          ;CLEAR ALL LOCATIONS OF TEST MAP
007502  016701  172302                  MOV     .TSTMAP,R1      ;TEST MAP TABLE
007506  005021                  1$:     CLR     (R1)+
007510  077002                          SOB     R0,1$           ;LOOP UNTIL DONE
007512  016700  172076                  MOV     SAVTST,R0       ;GET LOWER 128K MAP
007516  042700  177776                  BIC     #177776,R0      ;DO ONLY FIRST BANK
007522  010067  172026                  MOV     R0,TSTMAP       ;FOR DATA TESTS
007526                          2$:                             ;CONTINUE TESTING
```

```
                                .SBTTL SECTION 2:       WORST CASE NOISE TESTS
                                ;;*****************************************************************
                                ;*  THESE TESTS WRITE MEMORY WORST CASE NOISE TEST PATTERNS THROUGHOUT
                                ;*  MEMORY AND CHECK THAT THEY CAN BE WRITTEN AND READ.
                                ;;*****************************************************************
                                ;;*****************************************************************
                                ;*TEST 6        WRITE A CONSTANT INTO MEMORY.
                                ;*      THE CONSTANT IS USER SELECTABLE (DEFAULT = 0).
                                ;*      R0 = DATA WRITTEN INTO MEMORY (SHOULD BE)
                                ;*      R1 = DATA READ FROM MEMORY (WAS)
                                ;*      R2 = VIRTUAL ADDRESS
                                ;*      R3 = NOT USED
                                ;*      R4 = NOT USED
                                ;*      R5 = BLOCK BOUNDARY BIT MASK.
                                ;;*****************************************************************
007526                          TST6:
007526  004567  007622                  JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
007532  016700  172226          TST6A:  MOV     .CONST, R0      ;GET USER CONSTANT
007536  004467  003236                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
007542  010022                  1$:     MOV     R0,     (R2)+   ;WRITE CONSTANT INTO MEMORY.
007544  077502                          SOB     R5,     1$      ;BRANCH IF MORE IN CURRENT BLOCK
007546  004767  003440                  JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.

                                ;;*****************************************************************
                                ;*TEST 7        READ MEMORY AND COMPARE TO CONSTANT.
                                ;*  IMPORTANT:  THIS TEST SHOULD NOT BE RUN WITHOUT FIRST RUNNING TEST $TN.
                                ;;*****************************************************************
007552                          TST7:
007552  004567  007576                  JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
007556  016700  172202                  MOV     .CONST, R0      ;GET USER CONSTANT
007562  004467  003212                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
007566                          1$:
007566  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
007570  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
007572  001405                          BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
007574  004767  007024          64$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
007600  004767  010062                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
007604  000004                          .WORD   4               ;ERROR TYPE CODE.
007606                          65$:
007606  077511                          SOB     R5,     1$      ;BRANCH IF MORE IN CURRENT BLOCK
007610  004767  003376                  JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
                                ;*  SPECIAL CHECK TO SEE IF TEST 6 IS SELECTED THRU THE SWR.
                                ;*  ALLOWS THE OPERATOR TO SWITCH BACK AND FORTH BETWEEN TESTS 6 AND 7
                                ;*  BY SIMPLY "TOGGLING" SW00 WHEN SW01, SW02, AND SW08 ARE SET.
007614  032777  000400  171306          BIT     #SW08,  @SWR    ;CHECK THAT LOOP ON TEST BIT SET
007622  001416                          BEQ     TST10           ;BRANCH IF NOT LOOP ON TEST
007624  017746  171300                  MOV     @SWR,   -(SP)   ;GET SWITCH REGISTER DATA.
007630  042716  177740                  BIC     #177740,(SP)    ;CLEAR NON-TEST-NUMBER SWITCHS.
007634  022726  000006                  CMP     #6,     (SP)+   ;CHECK IF TEST 6 IN SWITCHES.
007640  001007                          BNE     TST10           ;BRANCH IF NOT TEST 6
007642  162767  000001  171222          SUB     #1,     $TSTNM  ;RESET TEST NUM
007650  162767  000024  171220          SUB     #TST7-TST6,$LPADR ;RESET LOOP ADR
007656  000725                          BR      TST6A           ;GO TO TEST 6

                                ;;*****************************************************************
                                ;*TEST 10       WORSE CASE NOISE (PARITY) WORD TESTING
```

```
2091                                        ;*  CHECK MEMORY WITH A SERIES OF PATTERNS
2092                                        ;;*************************************************************
2093  007660                                TST10:
2094  007660  004567  007470                       JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
2095  007664  016704  172136                       MOV     .MPPAT, R4      ;INITIALIZE PATTERN TABLE POINTER
2096  007670  004767  005766               1$:     JSR     PC,     CKPMER  ;CHECK FOR NON-TRAP PARITY MEMORY ERRORS.
2097  007674  012400                               MOV     (R4)+,  R0      ;GET THE DATA PATTERN.
2098  007676  001417                               BEQ     TST11           ;;BR IF END OF TABLE.
2099  007700  004467  003074                       JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2100  007704  010012                       2$:     MOV     R0,     (R2)    ;PUT DATA PATTERN INTO MEMORY.
2101  007706  012201                               MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2102  007710  020001                               CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2103  007712  001405                               BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2104  007714  004767  006704               64$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2105  007720  004767  007742                       JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2106  007724  000004                               .WORD   4               ;ERROR TYPE CODE.
2107  007726                               65$:
2108  007726  077512                               SOB     R5,     2$      ;BRANCH IF MORE IN CURRENT BLOCK
2109  007730  004767  003256                       JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 2$.
2110  007734  000755                               BR      1$              ;BR BACK TO DO NEXT PATTERN
```

```
                                ;;*****************************************************************
                                ;*TEST 11        ROTATE A "0" BIT THROUGH A FIELD OF ONES.
                                ;;*****************************************************************
007736                          TST11:
007736  004567  007412                  JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
007742  012700  177777                  MOV     #-1,    R0      ;SET CHECK WORD
007746  004767  004332                  JSR     PC,     SETCON  ;PUT THE CONTENTS OF R0 IN ALL MEMORY.
007752  004467  003022                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
007756  000241          1$:             CLC                     ;CLEAR CARRY BIT IN PSW
007760  004767  004336                  JSR     PC,     ROTATE
007764  016201  177776                  MOV     -2(R2), R1      ;GET RESULT
007770  103402                          BCS     63$             ;BRANCH IF 'C' BIT WAS SET
007772  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
007774  001405                          BEQ     64$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
007776  004767  006622          63$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
010002  004767  007660                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
010006  000005                          .WORD   5               ;ERROR TYPE CODE.
010010                          64$:
010010  077516                          SOB     R5,     1$      ;BRANCH IF MORE IN CURRENT BLOCK
010012  004767  003174                  JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.

                                ;;*****************************************************************
                                ;*TEST 12        ROTATE A "1" BIT THROUGH A FIELD OF ZEROS
                                ;;*****************************************************************
010016                          TST12:
010016  004567  007332                  JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
010022  005000                          CLR     R0              ;SET CHECK WORD
010024  004767  004254                  JSR     PC,     SETCON  ;PUT THE CONTENTS OF R0 IN ALL MEMORY
010030  004467  002744                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
010034  000261          1$:             SEC                     ;SET 'C' BIT IN PSW
010036  004767  004260                  JSR     PC,     ROTATE  ;GO ROTATE '1' BIT
010042  016201  177776                  MOV     -2(R2), R1      ;GET RESULT
010046  103002                          BCC     63$             ;BRANCH IF 'C' IS CLEAR
010050  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
010052  001405                          BEQ     64$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
010054  004767  006544          63$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
010060  004767  007602                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
010064  000005                          .WORD   5               ;ERROR TYPE CODE.
010066                          64$:
010066  077516                          SOB     R5,     1$      ;BRANCH IF MORE IN CURRENT BLOCK
010070  004767  003116                  JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
```

```
                                ;;*****************************************************************
                                ;*TEST 13       WORSE CASE NOISE PARITY BYTE TESTING
                                ;*  CHECK PARITY MEMORY WITH A SERIES OF BYTE PATTERNS
                                ;*       1) FORCE WRONG PARITY IN EACH BYTE OF PARITY MEMORY
                                ;*       2) READ IT BACK WITH ACTION ENABLE SET, MAKING SURE THAT A TRAP OCCURS
                                ;*       3) WRITE GOOD PARITY AND MAKE SURE NO TRAP OCCURS WHEN IT IS READ
                                ;*       4) MAKE SURE THE ERROR ADDRESS BITS (CSR BITS <11-5>) ARE CORRECT
                                ;*       5) IF MMU ENABLED AND ABOVE 128KW,CHECK BIT14 AND BITS <11-5>
                                ;*          ON PARITY CSR.
                                ;;*****************************************************************
010074                          TST13:
010074  004567  007254                  JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
010100  005767  173264          WWPB0:  TST     MPRX            ;CHECK FOR ANY PARITY MEMORY.
010104  001404                          BEQ     1$              ;BR IF NO PARITY MEMORY.
010106  032777  000100  171014          BIT     #SW06,  @SWR    ;CHECK FORINHIBIT PARITY SWITCH.
010114  001402                          BEQ     2$              ;BR IF NOT SET.
010116  000167  000742          1$:     JMP     TST14           ;SKIP THIS TEST IF NO PARITY MEMORY PRESENT.
010122  005000                  2$:     CLR     R0              ;ZERO TO BE PUT IN ALL MEMORY.
010124  004767  004154                  JSR     PC,     SETCON  ;ROUTINE TO LOAD ALL MEMORY.
010130  004467  002644                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.

                                                                ;RETURN FOR MMUP
010134  006305                          ASL     R5              ;MAKE TEST COUNTER BYTE VALUE
010136  012700  000020          WWPBYT: MOV     #20,    R0      ;TABLE COUNT
010142  016701  171646                  MOV     .BITPT, R1      ;BANK POINTER TABLE
010146  016703  171640                  MOV     .PMEMAP,R3      ;PARITY TABLE
010152  032123                  1$:     BIT     (R1)+,(R3)+     ;CHECK IF CURRENT BANK HAS PARITY MEMORY
010154  001004                          BNE     2$              ;BRANCH IF PARITY MEMORY
010156  077003                          SOB     R0,     1$      ;CHECK TILL END OF TABLE
010160  060502                          ADD     R5,     R2      ;IF NONE FOUND POINT R2 TO
                                                                ;FIRST ADDR OF NEXT BANK.
010162  000167  000666                  JMP     WWPB6           ;FIND NEXT BLOCK
010166  004767  005434          2$:     JSR     PC,     SETAE   ;SET ACTION ENABLE (EVEN IF BANK0.)
010172  004767  005464                  JSR     PC,     CKPMER  ;CHECK FOR ANY NON TRAP PARITY ERRORS.
010176  020227  000114          WWPB1:  CMP     R2,     #114    ;CHECK IF POINTING TO PARITY ERROR VECTOR.
010202  001004                          BNE     3$              ;BR IF NOT AT VECTOR.
010204  062702  000004                  ADD     #4,     R2      ;SKIP PARITY VECTOR.
010210  162705  000004                  SUB     #4,     R5      ;SKIP PARITY VECTOR
                                                                ;SUBTACT 4 FROM TEST COUNTER
010214  111201                  3$:     MOVB    (R2),   R1      ;CHECK IF BYTE STILL CLEARED.
010216  001405                          BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
010220  004767  006312          64$:    JSR     PC,     SPRNT   ;SET UP VALUES FOR ERROR PRINTING.
010224  004767  007436                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
010230  000011                          .WORD   11              ;ERROR TYPE CODE.
010232                          65$:
010232  105067  171522                  CLRB    OEFLG           ;CLEAR ODD/EVEN FLAG.
010236  112700  000252                  MOVB    #252,   R0      ;SET UP DATA...EVEN, SETS PARITY BIT.
010242  110012                  WWPB2:  MOVB    R0,     (R2)    ;MOV DATA INTO TEST LOCATION.
010244  016703  171552                  MOV     .MPRX,  R3      ;GET PARITY REGISTER TABLE POINTER.
010250  056773  171512  000000  10$:    BIS     WWP,@(R3)       ;SET WRITE WRONG PARITY.
010256  052733  000001                  BIS     #AE,@(R3)+
010262  005713                          TST     (R3)            ;CHECK FOR TABLE TERMINATOR.
010264  001371                          BNE     10$             ;BR IF MORE REGS IN TABLE.
                                ;*  SET WRONG PARITY IN LOCATION UNDER TEST.
010266  110012                          MOVB    R0,     (R2)    ;WRITE SAME DATA (EXCEPT PARITY) VIA DATOB.
010270  016703  171526                  MOV     .MPRX,  R3      ;GET PARITY REG TABLE POINTER.
```

```
 2209  010274  046733  171466          11$:    BIC     WWP,    @(R3)+  ;CLEAR WRITE WRONG PARITY.
 2210  010300  005713                          TST     (R3)            ;CHECK FOR TABLE TERMINATOR.
 2211  010302  001374                          BNE     11$             ;BR IF MORE PARITY REGISTERS.
 2212  010304  005767  171466                  TST     LSIFLG          ;CHECK IF RUNNING ON EARLIER LSI-11/2
 2213  010310  001402                          BEQ     12$             ;BR IF 11/23 OR LATER PROCESSOR
 2214  010312  105712                          TSTB    (R2)            ;PARITY TRAP LOGIC NOT AVAILABLE ON LSI11/2
 2215  010314  000424                          BR      PBTRP1          ;SET PARITY ERROR WITH READ AND BYPASS
 2216                                                                  ;PARITY TRAP AND TEST PARITY CSR.
 2217  010316  016737  171502  000114  12$:    MOV     .PBTRP, @#PARVEC ;SET UP VECTOR FOR EXPECTED TRAP.
 2218                                  ;*  DETECT WRONG PARITY VIA DATIO; DATOB SHOULDN'T EXECUTE.
 2219  010324  105412                          NEGB    (R2)            ;DATIO (DATOB AND COM PARITY BIT.)
 2220                                  ;*  SHOULD HAVE TRAPPED TO PBTRP.
 2221  010326  016737  171476  000114          MOV     .PESRV, @#PARVEC ;RESET VECTOR FOR UNEXPECTED TRAPS.
 2222  010334  004767  006240          64$:    JSR     PC,     SPRNTO  ;SET UP VALUES FOR ERROR PRINTING.
 2223  010340  004767  007322                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
 2224  010344  000012                          .WORD   12              ;ERROR TYPE CODE.
 2225  010346  004767  006142                  JSR     PC,CLRPAR       ;CLEAR ALL PARITY CSRS BEFORE NEXT BYTE
 2226  010352  000167  000442                  JMP     WWPB4           ;SKIP TRAP SERVICE.
 2227
 2228
 2229
 2230                                  ;*  EXPECTED PARITY MEMORY TRAPS COME HERE.
 2231  010356  016737  171446  000114  PBTRP:  MOV     .PESRV, @#PARVEC ;RESET PARITY VECTOR FOR UNEXPECTED TRAPS.
 2232  010364  022626                          CMP     (SP)+,  (SP)+   ;RESET THE STACK POINTER AFTER TRAP.
 2233  010366  016703  171426          PBTRP1: MOV     .MPR0,  R3      ;GET PARITY REG AND MAP TABLE POINTER.
 2234  010372  032713  000001          21$:    BIT     #BIT0,  (R3)    ;CHECK IF THIS REGISTER EXISTS.
 2235  010376  001003                          BNE     22$             ;BR IF IT DOESN'T EXIST.
 2236  010400  017301  000000                  MOV     @(R3),  R1      ;GET THE CONTENTS.
 2237  010404  100413                          BMI     23$             ;BR IF ERROR FLAG SET.
 2238  010406  062703  000044          22$:    ADD     #44,    R3      ;MOVE POINTER TO NEXT REG.
 2239  010412  020367  171404                  CMP     R3,     .MPRX   ;CHECK FOR END OF TABLE.
 2240  010416  103765                          BLO     21$             ;BR IF MORE REGISTERS.
 2241  010420  004767  006154          64$:    JSR     PC,     SPRNTO  ;SET UP VALUES FOR ERROR PRINTING.
 2242  010424  004767  007236                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
 2243  010430  000013                          .WORD   13              ;ERROR TYPE CODE.
 2244  010432  000572                          BR      WWPB4           ;EXIT AFTER ERROR.
 2245  010434  004567  005330          23$:    JSR     R5,     PARMAT  ;CHECK FOR MAP OF THIS REGISTER
 2246  010440  000405                          BR      24$             ;BRANCH IF THIS PARITY REGISTER
 2247                                                                  ;CONTROLS THIS BANK.
 2248                                                                  ;RETURN +2,ERROR ,CANNOT FIND MEM BANK
 2249                                                                  ;IN PARITY MAP TABLE.
 2250  010442  004767  006126          65$:    JSR     PC,     SPRNTP  ;SET UP VALUES FOR ERROR PRINTING.
 2251  010446  004767  007214                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
 2252  010452  000014                          .WORD   14              ;ERROR TYPE CODE.
 2253  010454                          24$:
 2254  010454  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
 2255  010456  010200                          MOV     R2,     R0      ;GET THE ADDRESS POINTER.
 2256  010460  042700  003777                  BIC     #3777,  R0      ;CLEAR LOW ADDRESS BITS.
 2257  010464  000300                          SWAB    R0              ;SHIFT 6 PLACES RIGHT.
 2258  010466  006300                          ASL     R0
 2259  010470  006300                          ASL     R0
 2260  010472  005767  170040                  TST     MMAVA           ;CHECK FOR MEM MGMT.
 2261  010476  001411                          BEQ     25$             ;BR IF NO MEM MGMT.
 2262  010500  042700  177600                  BIC     #177600,R0      ;CLEAR BANK BITS
 2263  010504  063700  172344                  ADD     @#KIPAR2,R0     ;ADD MEM MGMT OFFSET.
 2264  010510  032702  020000                  BIT     #BIT13, R2      ;CHECK FOR PAR3 REF. ADDRESSING
```

```
2265  010514  001402                        BEQ     25$             ;BR IF R2 IS ADDRESSING PAR2
2266  010516  062700  000200                ADD     #200,R0         ;SET UP EXPECTED WITH PAR3
2267  010522  052700  100001        25$:    BIS     #BIT15+BIT0,R0  ;SET ERROR AND AE BIT IN CHECK WORD.
2268  010526  016367  000042  170752        MOV     42(R3), RESRVD  ;GET APPROPIATE MASK.
2269  010534  046700  170746                BIC     RESRVD, R0      ;CLEAR PARITY REG BITS RESERVED FOR FUTURE.
2270  010540  046701  170742                BIC     RESRVD, R1      ;CLEAR PARITY REG BITS RESERVED FOR FUTURE.
2271                                ;NOTE: THE ABOVE INSTRUCTION (2 WORDS) CAN BE NOP'ED FOR UNMIXED MEMORY TYPES.
2272  010544  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2273  010546  001405                        BEQ     67$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2274  010550  004767  006020        66$:    JSR     PC,     SPRNTP  ;SET UP VALUES FOR ERROR PRINTING.
2275  010554  004767  007106                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2276  010560  000015                        .WORD   15              ;ERROR TYPE CODE.
2277  010562                        67$:
2278  010562  005767  167750                TST     MMAVA           ;CHECK FOR 22 BIT ADDRESSING
2279  010566  100034                        BPL     100$            ;BR IF 22 BIT ADDR NOT ENABLED
2280  010570  022737  010000  172344        CMP     #10000,@#KIPAR2 ;CHECK IF ABOVE 128KW
2281  010576  101030                        BHI     100$            ;BR IF KIPAR2 LESS THAN 128KW
2282  010600  010046                        MOV     R0,-(SP)        ;;PUSH R0 ON STACK
2283  010602  052773  040000  000000        BIS     #BIT14,@(R3)    ;SET BIT14 TO ENABLE BITS 18-21
2284                                                                ;INTO BITS 8-5 OF PARITY CSR
2285  010610  013700  172344                MOV     @#KIPAR2,R0     ;SETUP EXPECTED FOR PARITY CSR
2286  010614  042700  007777                BIC     #7777,R0        ;BITS 18-21 STORED IN
2287  010620  000300                        SWAB    R0              ;BITS 5-8.
2288  010622  006300                        ASL     R0              ;A18-21 IN BITS 5-8
2289  010624  052700  140001                BIS     #BIT15!BIT14!AE,R0;SAVE EXPECTED PARITY CSR
2290  010630  017301  000000                MOV     @(R3),R1        ;GET CONTENTS OF PARITY CSR
2291  010634  042701  030032                BIC     #30032,R1       ;CLEAR MASKED BITS
2292  010640  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2293  010642  001405                        BEQ     69$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2294  010644  004767  005724        68$:    JSR     PC,     SPRNTP  ;SET UP VALUES FOR ERROR PRINTING.
2295  010650  004767  007012                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2296  010654  000015                        .WORD   15              ;ERROR TYPE CODE.
2297  010656                        69$:
2298  010656  012600                        MOV     (SP)+,R0        ;;POP STACK INTO R0
2299  010660  005073  000000        100$:   CLR     @(R3)           ;CLEAR REG INCLUDING ACTION ENABLE.
2300  010664  010346                        MOV     R3,-(SP)        ;;PUSH R3 ON STACK
2301  010666  062703  000044        26$:    ADD     #44,    R3      ;UPDATE POINTER TO NEXT PARITY REG + MAP.
2302  010672  020367  171124                CMP     R3,     .MPRX   ;CHECK FOR END OF TABLE.
2303  010676  101014                        BHI     WWPB3           ;BR IF END OF TABLE REACHED.
2304  010700  032713  000001                BIT     #BIT0,  (R3)    ;CHECK IF NEXT REG EXISTS.
2305  010704  001370                        BNE     26$             ;BR IF THIS PARITY REG DOESN'T EXIST.
2306  010706  017301  000000                MOV     @(R3),  R1      ;SAVE AND CHECK FOR ERROR FLAG.
2307  010712  100365                        BPL     26$             ;BR IF NO ERROR FLAG.
2308  010714  004767  005654        70$:    JSR     PC,     SPRNTP  ;SET UP VALUES FOR ERROR PRINTING.
2309  010720  004767  006742                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2310  010724  000016                        .WORD   16              ;ERROR TYPE CODE.
2311  010726  000757                        BR      26$             ;BR AFTER ERROR.
2312  010730  111204                WWPB3:  MOVB    (R2),   R4      ;GET THE DATA FOR CHECKING.
2313                                ;* READING THE DATA VIA DATI TO CHECK IT SHOULD CAUSE PARITY ERROR, BUT
2314                                ;*  ACTION ENABLE IS NOT SET IN CONTROLLING REG, SO NO TRAP SHOULD OCCUR.
2315  010732  111212                        MOVB    (R2),   (R2)    ;RESTORE RIGHT PARITY
2316                                ;NOTE:  THE ABOVE INSTRUCTION CAN BE NOP'ED FOR PROCESSORS
2317                                ;       WHICH DO ONLY DATOB TO DESTINATION OF MOVB INSTRUCTIONS.
2318  010734  012603                        MOV     (SP)+,R3        ;;POP STACK INTO R3
2319  010736  017301  000000                MOV     @(R3),  R1      ;READ THE PARITY REGISTER TO CHECK IT AGAIN.
2320  010742  046701  170540                BIC     RESRVD, R1      ;CLEAR PARITY REG BITS RESERVED FOR FUTURE.
```

```
2321                                            ;NOTE: THE ABOVE INSTRUCTION (2 WORDS) CAN BE NOP'ED FOR UNMIXED MEMORY TYPES.
2322  010746  042700  000001                            BIC     #AE,    R0      ;CLEAR THE ACTION ENABLE BIT IN TEST DATA.
2323  010752  020001                                    CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2324  010754  001405                                    BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2325  010756  004767  005612                    64$:    JSR     PC,     $PRNTP  ;SET UP VALUES FOR ERROR PRINTING.
2326  010762  004767  006700                            JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2327  010766  000015                                    .WORD   15              ;ERROR TYPE CODE.
2328  010770                                    65$:
2329  010770  012773  000001  000000                    MOV     #1,     @(R3)   ;CLEAR ALL BUT ACTION ENABLE.
2330  010776  010401                                    MOV     R4,     R1      ;GET DATA READ FROM MEMORY FOR TESTING.
2331  011000  012600                                    MOV     (SP)+,R0        ;;POP STACK INTO R0
2332  011002  120001                                    CMPB    R0,     R1      ;CHECK THE DATA.
2333  011004  001405                                    BEQ     67$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2334  011006  004767  005566                    66$:    JSR     PC,     $PRNT0  ;SET UP VALUES FOR ERROR PRINTING.
2335  011012  004767  006650                            JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2336  011016  000017                                    .WORD   17              ;ERROR TYPE CODE.
2337  011020                                    67$:
2338  011020  110012                            WWPB4:  MOVB    R0,     (R2)    ;RESTORE DATA.
2339  011022  105712                                    TSTB    (R2)            ;DO A DATI TO BE SURE RIGHT PARITY.
2340  011024  012700  000253                            MOV     #253,   R0      ;SET ODD PARITY DATA.
2341  011030  105167  170724                            COMB    OEFLG           ;CHECK IF DONE BOTH ODD AND EVEN PARITY.
2342  011034  100002                                    BPL     27$             ;BR IF DONE BOTH EVEN AND ODD.
2343  011036  000167  177200                            JMP     WWPB2           ;LOOP BACK AND DO ODD(PARITY BIT CLR).
2344  011042  005202                            27$:    INC     R2              ;MOVE POINTER TO NEXT MEMORY BYTE.
2345  011044  005305                            WWPB5:  DEC     R5              ;CHECK FOR END OF BLOCK
2346  011046  001402                                    BEQ     WWPB6           ;BR IF END OF BLOCK FOUND.
2347  011050  000167  177122                            JMP     WWPB1           ;LOOP BACK TO TEST NEXT BYTE.
2348  011054  004767  002132                    WWPB6:  JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO WWPBYT
2349  011060  004767  004476                            JSR     PC,     MAMF    ;GO RESET PARITY REGISTERS.
```

```
                                ;;*****************************************************************
                                ;*TEST 14       RANDOM DATA  TESTING THRU PROGRAM CODE RELOCATION.
                                ;;*****************************************************************
011064                          TST14:
011064  004567  006264                JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
011070  010703                RANTST: MOV     PC,     R3      ;GET CURRENT PROGRAM COUNTER.
011072  042703  007777                BIC     #7777,  R3      ;POINT TO BEGINNING OF CURRENT 2K BLOCK.
011076  004467  001676                JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
011102  010246                1$:     MOV     R2,     -(SP)   ;SAVE MEMORY POINTER.
011104  010346                        MOV     R3,     -(SP)   ;SAVE "DATA" POINTER.
011106  010567  170660                MOV     R5,     TEMP1   ;SAVE ADDRESS COUNTER
011112  012322                2$:     MOV     (R3)+,  (R2)+   ;MOV CODE INTO TEST MEMORY.
011114  032703  007777                BIT     #7777,  R3      ;CHECK FOR END OF "DATA TABLE"
011120  001002                        BNE     3$              ;BRANCH IF MORE
011122  162703  010000                SUB     #10000, R3      ;RESET POINTER TO START OF "RANDOM DATA"
011126  077507                3$:     SOB     R5,     2$      ;IF NOT END OF BLOCK,BRANCH TO 2$
011130  012603                        MOV     (SP)+,  R3      ;RESET "DATA" POINTER.
011132  012602                        MOV     (SP)+,  R2      ;RESET MEMORY POINTER.
011134  016705  170632                MOV     TEMP1,  R5      ;RESET ADDRESS COUNTER
011140  012300                4$:     MOV     (R3)+,  R0      ;GET S/B DATA.
011142  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
011144  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
011146  001405                        BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
011150  004767  005450        64$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
011154  004767  006506                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
011160  000020                        .WORD   20              ;ERROR TYPE CODE.
011162                          65$:
011162  032703  007777                BIT     #7777,  R3      ;CHECK FOR END OF "DATA TABLE"
011166  001002                        BNE     5$              ;BR IF MORE.
011170  162703  010000                SUB     #10000, R3      ;RESET POINTER TO TOP OF "DATA TABLE".
011174                          5$:
011174  077517                        SOB     R5,     4$      ;BRANCH IF MORE IN CURRENT BLOCK
011176  004767  002010                JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.

                                .SBTTL SECTION 3:       INSTRUCTION EXECUTION TESTS.
                                ;;*****************************************************************
                                ;*TEST 15       EXECUTE DATI, DATO THRU MEMORY.
                                ;*  EXECUTES THE INSTRUCTION "MOV R4,(R2)" THROUGHOUT MEMORY.
                                ;*  AN "RTS R5" (CODE 205) IS PLACED AFTER THE "MOV" INSTRUCTION TO RETURN
                                ;*  CONTROL TO THE MAIN PROGRAM FOR INSTRUCTION EXECUTION CHECKOUT.
                                ;*  THIS IS AN EXAMPLE OF WHAT THIS TEST DOES IN RELATION TO MEMORY:
                                ;*
                                ;*                MEMORY        INSTRUCTION     CONTENTS OF MEMORY LOCATION
                                ;*                LOCATION      PLACED THERE    AFTER INSTRUCTION EXECUTION
                                ;*
                                ;*   1ST PASS /   40000           010412          000205
                                ;*  THRU TEST /   40002           000205          000205
                                ;*
                                ;*   2ND PASS /   40002           010412          000205
                                ;*  THRU TEST /   40004           000205          000205
                                ;*
                                ;*      ETC., ETC., ETC.
                                ;*
                                ;*  R0 - DATA WRITTEN ON TOP OF IUT BY THE IUT (SHOULD BE).
                                ;*  R1 = DATA READ FROM MEMORY (WAS).
                                ;*  R2 - ADDRESS OF IUT/DATA.
```

```
2406                                    ;*  R3 = INSTRUCTION UNDER TEST (IUT).
2407                                    ;*  R4 = RTS R5 (CODE 205).
2408                                    ;*  R5 = BLOCK BOUNDARY BIT MASK.
2409                                    ;;*****************************************************************
2410  011202                            TST15:
2411  011202  004567  006146                   JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
2412  011206  012703  010412           DIDO:   MOV     #010412,R3      ;GET 'MOV R4,(R2)' INSTRUCTION (IUT).
2413  011212  012704  000205                   MOV     #205,   R4      ;GET 'RTS R5'
2414  011216  010400                           MOV     R4,     R0      ;SET UP S/B DATA AFTER EXECUTION.
2415  011220  004467  001554                   JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2416  011224  005305           1$:     DEC     R5                      ;DO ALL ADDRESSES -1
2417  011226  010322                           MOV     R3,     (R2)+   ;PUT IUT INTO FIRST LOC OF BLOCK.
2418  011230  010412           2$:     MOV     R4,     (R2)    ;PUT 'RTS R5' FOLLOWING IUT.
2419  011232  004542                   JSR     R5,     -(R2)   ;GO EXECUTE THE IUT.
2420  011234  012201                   MOV     (R2)+,  R1      ;GET THE DATA FROM THE MEM ADR UNDER TEST.
2421  011236  020001                   CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2422  011240  001405                   BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2423  011242  004767  005352           64$:    JSR     PC,     SPRNT3  ;SET UP VALUES FOR ERROR PRINTING.
2424  011246  004767  006414                   JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2425  011252  000021                           .WORD   21              ;ERROR TYPE CODE.
2426  011254                           65$:
2427  011254  010322                           MOV     R3,     (R2)+   ;PUT THE IUT INTO THE NEXT LOCATION.
2428  011256  077514                           SOB     R5,     2$      ;BRANCH IF MORE IN CURRENT BLOCK.
2429  011260  004767  001726                   JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
```

```
                                ;;*****************************************************************
                                ;*TEST 16       EXECUTE DATI, DATOB (LOW BYTE) THRU MEMORY.
                                ;*  EXECUTES THE INSTRUCTION 'MOVB R4,(R2)' THROUGHOUT MEMORY.
                                ;*  AN 'RTS R5' (CODE 205) IS PLACED AFTER THE 'MOVB' INSTRUCTION TO RETURN
                                ;*  CONTROL TO THE MAIN PROGRAM FOR INSTRUCTION EXECUTION CHECKOUT.
                                ;*  THIS IS AN EXAMPLE OF WHAT THIS TEST DOES IN RELATION TO MEMORY:
                                ;*
                                ;*                MEMORY       INSTRUCTION     CONTENTS OF MEMORY LOCATION
                                ;*               LOCATION      PLACED THERE    AFTER INSTRUCTION EXECUTION
                                ;*
                                ;*   1ST PASS /  40000          110412          110605
                                ;*  THRU TEST /  40002          000205          000205
                                ;*
                                ;*   2ND PASS /  40002          110412          110605
                                ;*  THRU TEST /  40004          000205          000205
                                ;*
                                ;*      ETC., ETC., ETC.
                                ;*
                                ;*  R0 = DATA WRITTEN ON TOP OF IUT BY THE IUT (SHOULD BE).
                                ;*  R1 = DATA READ FROM MEMORY (WAS).
                                ;*  R2 = ADDRESS OF IUT/DATA.
                                ;*  R3 = INSTRUCTION UNDER TEST (IUT).
                                ;*  R4 = RTS R5 (CODE 205).
                                ;*  R5 = BLOCK BOUNDARY BIT MASK.
                                ;*****************************************************************
011264                          TST16:
011264  004567  006064                  JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
011270  012703  110412          DIDBL:  MOV     #110412,R3      ;GET 'MOVB R4,(R2)' INSTRUCTION (IUT).
011274  012704  000205                  MOV     #205,   R4      ;GET 'RTS R5'
011300  012700  110605                  MOV     #110605,R0      ;SET UP S/B DATA AFTER EXECUTION.
011304  004467  001470                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
011310  005305          1$:     DEC     R5                      ;DO ALL ADDRESSES -1
011312  010322                  MOV     R3,     (R2)+   ;PUT IUT INTO FIRST LOC OF BLOCK.
011314  010412          2$:     MOV     R4,     (R2)    ;PUT 'RTS R5' FOLLOWING IUT.
011316  004542                  JSR     R5,     -(R2)   ;GO EXECUTE THE IUT.
011320  012201                  MOV     (R2)+,  R1      ;GET THE DATA FROM THE MEM ADR UNDER TEST.
011322  020001                  CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
011324  001405                  BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
011326  004767  005266          64$:    JSR     PC,     SPRNT3  ;SET UP VALUES FOR ERROR PRINTING.
011332  004767  006330                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
011336  000021                  65$:    .WORD   21              ;ERROR TYPE CODE.
011340
011340  010322                  MOV     R3,     (R2)+   ;PUT THE IUT INTO THE NEXT LOCATION.
011342  077514                  SOB     R5,     2$      ;BRANCH IF MORE IN CURRENT BLOCK.
011344  004767  001642                  JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
```

```
 2475                                   ;;*****************************************************************
 2476                                   ;*TEST 17       EXECUTE DATI, DATOB (HIGH BYTE) THRU MEMORY.
 2477                                   ;*  EXECUTES THE INSTRUCTION 'MOVB R3,-(R2)' THROUGHOUT MEMORY.
 2478                                   ;*  AN 'RTS R5' (CODE 205) IS PLACED AFTER THE 'MOVB' INSTRUCTION TO RETURN
 2479                                   ;*  CONTROL TO THE MAIN PROGRAM FOR INSTRUCTION EXECUTION CHECKOUT.
 2480                                   ;*  THIS IS AN EXAMPLE OF WHAT THIS TEST DOES IN RELATION TO MEMORY:
 2481                                   ;*
 2482                                   ;*                MEMORY        INSTRUCTION     CONTENTS OF MEMORY LOCATION
 2483                                   ;*               LOCATION       PLACED THERE    AFTER INSTRUCTION EXECUTION
 2484                                   ;*
 2485                                   ;*   1ST PASS /  40000           110342          161342
 2486                                   ;*  THRU TEST /  40002           000205          000205
 2487                                   ;*
 2488                                   ;*   2ND PASS /  40002           110342          161342
 2489                                   ;*  THRU TEST /  40004           000205          000205
 2490                                   ;*
 2491                                   ;*      ETC., ETC., ETC.
 2492                                   ;*
 2493                                   ;*  R0 = DATA WRITTEN ON TOP OF IUT BY THE IUT (SHOULD BE).
 2494                                   ;*  R1 = DATA READ FROM MEMORY (WAS).
 2495                                   ;*  R2 = ADDRESS OF IUT/DATA.
 2496                                   ;*  R3 = INSTRUCTION UNDER TEST (IUT).
 2497                                   ;*  R4 = RTS R5 (CODE 205).
 2498                                   ;*  R5 = BLOCK BOUNDARY BIT MASK.
 2499                                   ;*****************************************************************
 2500  011350                           1ST17:
 2501  011350  004567  006000                  JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
 2502  011354  012703  110342          DIDBH:  MOV     #110342,R3      ;GET 'MOVB R3,-(R2)' INSTRUCTION (IUT).
 2503  011360  012704  000205                  MOV     #205,   R4      ;GET 'RTS R5'
 2504  011364  012700  161342                  MOV     #161342,R0      ;SET UP S/B DATA AFTER EXECUTION.
 2505  011370  004467  001404                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
 2506  011374  005305                  1$:     DEC     R5              ;DO ALL ADDRESSES -1
 2507  011376  010322                          MOV     R3,     (R2)+   ;PUT IUT INTO FIRST LOC OF BLOCK.
 2508  011400  010412                  2$:     MOV     R4,     (R2)    ;PUT 'RTS R5' FOLLOWING IUT.
 2509  011402  004562  177776                  JSR     R5,     -2(R2)  ;GO EXECUTE THE IUT.
 2510  011406  005302                          DEC     R2              ;ADJUST R2 TO POINT TO MAUT.
 2511  011410  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM THE MEM ADR UNDER TEST.
 2512  011412  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
 2513  011414  001405                          BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
 2514  011416  004767  005176          64$:    JSR     PC,     SPRNT3  ;SET UP VALUES FOR ERROR PRINTING.
 2515  011422  004767  006240                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
 2516  011426  000021                          .WORD   21              ;ERROR TYPE CODE.
 2517  011430                           65$:
 2518  011430  010322                          MOV     R3,     (R2)+   ;PUT THE IUT INTO THE NEXT LOCATION.
 2519  011432  077516                          SOB     R5,     2$      ;BRANCH IF MORE IN CURRENT BLOCK.
 2520  011434  004767  001552                  JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
```

```
                                        ;;*****************************************************************
                                        ;*TEST 20       EXECUTE DATI, DATIO, DATO THRU MEMORY.
                                        ;*  EXECUTES THE INSTRUCTION 'NEG (R2)' THROUGHOUT MEMORY.
                                        ;*  AN 'RTS R5' (CODE 205) IS PLACED AFTER THE 'NEG' INSTRUCTION TO RETURN
                                        ;*  CONTROL TO THE MAIN PROGRAM FOR INSTRUCTION EXECUTION CHECKOUT.
                                        ;*  THIS IS AN EXAMPLE OF WHAT THIS TEST DOES IN RELATION TO MEMORY:
                                        ;*
                                        ;*                MEMORY       INSTRUCTION     CONTENTS OF MEMORY LOCATION
                                        ;*               LOCATION      PLACED THERE    AFTER INSTRUCTION EXECUTION
                                        ;*
                                        ;*   1ST PASS /  40000          005412          172366
                                        ;*  THRU TEST /  40002          000205          000205
                                        ;*
                                        ;*   2ND PASS /  40002          005412          172366
                                        ;*  THRU TEST /  40004          000205          000205
                                        ;*
                                        ;*      ETC., ETC., ETC.
                                        ;*

                                        ;*  R0 = DATA WRITTEN ON TOP OF IUT BY THE IUT (SHOULD BE).
                                        ;*  R1 = DATA READ FROM MEMORY (WAS).
                                        ;*  R2 = ADDRESS OF IUT/DATA.

                                        ;*  R3 = INSTRUCTION UNDER TEST (IUT).
                                        ;*  R4 = RTS R5 (CODE 205).
                                        ;*  R5 - BLOCK BOUNDARY BIT MASK.
                                        ;;*****************************************************************
011440                                  TST20:
011440  004567  005710                          JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
011444  012703  005412                  DIPDO:  MOV     #005412,R3      ;GET 'NEG (R2)' INSTRUCTION (IUT).
011450  012704  000205                          MOV     #205,   R4      ;GET 'RTS R5'
011454  012700  172366                          MOV     #172366,R0      ;SET UP S/B DATA AFTER EXECUTION.
011460  004467  001314                          JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
011464  005305                          1$:     DEC     R5              ;DO ALL ADDRESSES -1
011466  010322                          2$:     MOV     R3,     (R2)+   ;PUT IUT INTO FIRST LOC OF BLOCK.
011470  010412                                  MOV     R4,     (R2)    ;PUT 'RTS R5' FOLLOWING IUT.
011472  004542                                  JSR     R5,     -(R2)   ;GO EXECUTE THE IUT.
011474  012201                                  MOV     (R2)+,  R1      ;GET THE DATA FROM THE MEM ADR UNDER TEST.
011476  020001                                  CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
011500  001405                                  BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
011502  004767  005112                  64$:    JSR     PC,     SPRNT3  ;SET UP VALUES FOR ERROR PRINTING.
011506  004767  006154                          JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
011512  000021                                  .WORD   21              ;ERROR TYPE CODE.
011514                                  65$:
011514  010322                                  MOV     R3,     (R2)+   ;PUT THE IUT INTO THE NEXT LOCATION.
011516  077514                                  SOB     R5,     2$      ;BRANCH IF MORE IN CURRENT BLOCK.
011520  004767  001466                          JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
```

```
                                ;;*****************************************************************
                                ;*TEST 21       EXECUTE DATI, DATI, DATIO, DATOB (LOW BYTE) THRU MEMORY.
                                ;*  EXECUTES THE INSTRUCTION BICB (R2)+,-(R2)' THROUGHOUT MEMORY.
                                ;*  AN 'RTS R5' (CODE 205) IS PLACED AFTER THE BICB' INSTRUCTION TO RETURN
                                ;*  CONTROL TO THE MAIN PROGRAM FOR INSTRUCTION EXECUTION CHECKOUT.
                                ;*  THIS IS AN EXAMPLE OF WHAT THIS TEST DOES IN RELATION TO MEMORY:
                                ;*
                                ;*                MEMORY       INSTRUCTION     CONTENTS OF MEMORY LOCATION
                                ;*               LOCATION      PLACED THERE    AFTER INSTRUCTION EXECUTION
                                ;*
                                ;*   1ST PASS /  40000          142242          142000
                                ;*  THRU TEST /  40002          000205          000205
                                ;*
                                ;*   2ND PASS /  40002          142242          142000
                                ;*  THRU TEST /  40004          000205          000205
                                ;*
                                ;*      ETC., ETC., ETC.
                                ;*
                                ;*  R0 = DATA WRITTEN ON TOP OF IUT BY THE IUT (SHOULD BE).
                                ;*  R1 = DATA READ FROM MEMORY (WAS).
                                ;*  R2 = ADDRESS OF IUT/DATA.
                                ;*  R3 = INSTRUCTION UNDER TEST (IUT).
                                ;*  R4 = RTS R5 (CODE 205).
                                ;*  R5 = BLOCK BOUNDARY BIT MASK.
                                ;*****************************************************************
011524                          TST21:
011524  004567  005624                  JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
011530  012703  142242          DPDBL:  MOV     #142242,R3      ;GET BICB (R2)+,-(R2)' INSTRUCTION (IUT).
011534  012704  000205                  MOV     #205,   R4      ;GET 'RTS R5'
011540  012700  142000                  MOV     #142000,R0      ;SET UP S/B DATA AFTER EXECUTION.
011544  004467  001230                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
011550  005305          1$:     DEC     R5                      ;DO ALL ADDRESSES -1
011552  010322                  MOV     R3,     (R2)+   ;PUT IUT INTO FIRST LOC OF BLOCK.
011554  010412          2$:     MOV     R4,     (R2)    ;PUT 'RTS R5' FOLLOWING IUT.
011556  004542                  JSR     R5,     -(R2)   ;GO EXECUTE THE IUT.
011560  012201                  MOV     (R2)+,  R1      ;GET THE DATA FROM THE MEM ADR UNDER TEST.
011562  020001                  CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
011564  001405                  BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
011566  004767  005026          64$:    JSR     PC,     SPRNT3  ;SET UP VALUES FOR ERROR PRINTING.
011572  004767  006070                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
011576  000021                  .WORD   21              ;ERROR TYPE CODE.
011600                          65$:
011600  010322                  MOV     R3,     (R2)+   ;PUT THE IUT INTO THE NEXT LOCATION.
011602  077514                  SOB     R5,     2$      ;BRANCH IF MORE IN CURRENT BLOCK.
011604  004767  001402                  JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
```

```
2613                                    ;;*****************************************************************
2614                                    ;*TEST 22       EXECUTE DATI, DATI, DATIO, DATOB (HIGH BYTE) THRU MEMORY.
2615                                    ;*  EXECUTES THE INSTRUCTION 'BISB (R2)+,(R2)' THROUGHOUT MEMORY.
2616                                    ;*  AN 'RTS R5' (CODE 205) IS PLACED AFTER THE 'BISB' INSTRUCTION TO RETURN
2617                                    ;*  CONTROL TO THE MAIN PROGRAM FOR INSTRUCTION EXECUTION CHECKOUT.
2618                                    ;*  THIS IS AN EXAMPLE OF WHAT THIS TEST DOES IN RELATION TO MEMORY:
2619                                    ;*
2620                                    ;*                MEMORY       INSTRUCTION     CONTENTS OF MEMORY LOCATION
2621                                    ;*                LOCATION     PLACED THERE    AFTER INSTRUCTION EXECUTION
2622                                    ;*
2623                                    ;*   1ST PASS /   40000          152212          157212
2624                                    ;*  THRU TEST /   40002          000205          000205
2625                                    ;*
2626                                    ;*   2ND PASS /   40002          152212          157212
2627                                    ;*  THRU TEST /   40004          000205          000205
2628                                    ;*
2629                                    ;*      ETC., ETC., ETC.
2630                                    ;*
2631                                    ;*  R0 = DATA WRITTEN ON TOP OF IUT BY THE IUT (SHOULD BE).
2632                                    ;*  R1 - DATA READ FROM MEMORY (WAS).
2633                                    ;*  R2 - ADDRESS OF IUT/DATA.
2634                                    ;*  R3 = INSTRUCTION UNDER TEST (IUT).
2635                                    ;*  R4 = RTS R5 (CODE 205).
2636                                    ;*  R5 = BLOCK BOUNDARY BIT MASK.
2637                                    ;*****************************************************************
2638  011610                            1ST22:
2639  011610  004567  005540                   JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
2640  011614  012703  152212           DPDBH:  MOV     #152212,R3      ;GET 'BISB (R2)+,(R2)' INSTRUCTION (IUT).
2641  011620  012704  000205                   MOV     #205,   R4      ;GET 'RTS R5'
2642  011624  012700  157212                   MOV     #157212,R0      ;SET UP S/B DATA AFTER EXECUTION.
2643  011630  004467  001144                   JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2644  011634  005305                   1$:     DEC     R5              ;DO ALL ADDRESSES -1
2645  011636  010322                           MOV     R3,     (R2)+   ;PUT IUT INTO FIRST LOC OF BLOCK.
2646  011640  010412                   2$:     MOV     R4,     (R2)    ;PUT 'RTS R5' FOLLOWING IUT.
2647  011642  004542                           JSR     R5,     -(R2)   ;GO EXECUTE THE IUT.
2648  011644  005302                           DEC     R2              ;RESET R2 TO POINT TO IUT.
2649  011646  012201                           MOV     (R2)+,  R1      ;GET THE DATA FROM THE MEM ADR UNDER TEST.
2650  011650  020001                           CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2651  011652  001405                           BEQ     6$              ;BRANCH OVER ERROR CALL IF GOOD DATA.
2652  011654  004767  004740           64$:    JSR     ..,     SPRNT3  ;SET UP VALUES FOR ERROR PRINTING.
2653  011660  004767  006002                   JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2654  011664  000021                           .WORD   21              ;ERROR TYPE CODE.
2655  011666                           65$:
2656  011666  010322                           MOV     R3,     (R2)+   ;PUT THE IUT INTO THE NEXT LOCATION.
2657  011670  077515                           SOB     R5,     2$      ;BRANCH IF MORE IN CURRENT BLOCK.
2658  011672  004767  001314                   JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
```

```
                                .SBTTL SECTION 4:MOS TESTS
                                ;;*****************************************************************
                                ;*TEST 23       MARCHING 1'S AND 0'S.
                                ;* THIS TEST IS DESIGNED TO STRESS MOS MEMORIES.
                                ;* STARTING AT THE BOTTOM ADDRESS AND ADDRESSING UPWARDS A 8K BANK IS
                                ;* WRITTEN WITH 000377.THEN STARTING AT THE TOP ADDRESS OF THE BANK THE
                                ;* 000377 IS READ,THE BYTES ARE SWAPPED TO 177400 AND THE LOCATION
                                ;* REREAD TO CONFIRM THE WRITE.THIS IS REPEATED FOR EVERY LOCATION
                                ;* ADDRESSED DOWNWARD UNTIL THE BOTTOM IS REACHED. STARTING AT THE
                                ;* BOTTOM EACH LOCATION IS READ FOR 177400,THE BYTES ARE SWAPPED TO
                                ;* 000377 AND REREAD TO CONFIRM THE WRITE UNTIL THE TOP ADDRESS OF THE
                                ;* BANK IS REACHED. AGAIN STARTING AT THE BOTTOM EACH LOCATION IS READ
                                ;* FOR 000377,THE BYTES SWAPPED TO 177400 AND THE LOCATION REREAD TO
                                ;* CONFIRM THE WRITE. LASTLY STARTING FROM THE TOP AND ADDRESSING DOWN-
                                ;* WARD EACH LOCATION IS READ,THE BYTES SWAPPED TO 000377 AND THE
                                ;* LOCATION IS REREAD TO CONFIRM THE WRITE. THIS IS REPEATED FOR EVERY
                                ;*8K BANK UNDER TEST.
                                ;*
                                ;*      R0=DATA WRITTEN INTO MEMORY(SHOULD BE)
                                ;*      R1=DATA READ FROM MEMORY(WAS)
                                ;*      R2=VIRTUAL ADDRESS
                                ;*      R3=TIMES THROUGH COUNTER
                                ;*      R4=NOT USED
                                ;*      R5=BLOCK BOUNDARY BIT MASK.
                                ;;*****************************************************************
011676                          TST23:
011676  004567  005452                JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
011702  004467  001072                JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
011706  010567  170060          100$: MOV     R5,TEMP1        ;STORE TEST WORD COUNTER
011712  010267  170052                MOV     R2,TEMP         ;SAVE BANK STARTING ADDRESS
011716  005003                  1$:   CLR     R3              ;CLEAR PASS COUNTER
011720  012700  000377                MOV     #000377,R0      ;SETUP TO WRITE PATTERN
011724  010022                  2$:   MOV     R0,(R2)+        ;WRITE PATTERN
011726  077502                        SOB     R5,2$           ;END OF 8K?
011730  016705  170036                MOV     TEMP1,R5        ;RESTORE TEST WORD COUNTER FOR NEXT PASS
011734  014201                  3$:   MOV     -(R2),R1        ;GET DATA WRITEN
011736  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
011740  001405                        BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
011742  004767  004656          64$:  JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
011746  004767  005714                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
011752  000010                        .WORD   10              ;ERROR TYPE CODE.
011754                          65$:
011754  000300                  4$:   SWAB    R0              ;SWAP BYTES OF DATA
011756  010012                        MOV     R0,(R2) ;WRITE SWAPPED WORD
011760  011201                        MOV     (R2),R1         ;GET DATA WRITEN
011762  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
011764  001405                        BEQ     67$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
011766  004767  004632          66$:  JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
011772  004767  005670                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
011776  000010                        .WORD   10              ;ERROR TYPE CODE.
012000                          67$:
012000  000300                        SWAB    R0              ;PUT DATA BACK TO ORINGINAL
012002  005703                        TST     R3              ;IF ON PASS 0 OR PASS 3
012004  001403                        BEQ     5$              ;WE ARE ADDRESSING DOWN
012006  020327  000003                CMP     R3,#3           ;IF ON PASS 1 OR 2 GO TO
012012  001011                        BNE     6$              ;UPWARD
```

```
2715  012014  077531           5$:     SOB     R5,3$           ;DONE A PASS? ,IF NOT THEN 3$
2716  012016  005203                    INC     R3              ;IF YES INCREMENT PASS COUNTER
2717  012020  022703  000004           CMP     #4,R3           ;ARE WE DONE ALL PASSES FOR THIS 8K?
2718  012024  001433                    BEQ     9$              ;IF YES BRANCH
2719  012026  016705  167740           MOV     TEMP1,R5        ;RESTORE TEST COUNTER FOR NEXT PASS.
2720  012032  000300                    SWAB    R0              ;ELSE SET UP NEW READ WORD
2721  012034  000404                    BR      7$              ;GO TO START OF ADDRESS UP
2722  012036  062702  000002     6$:    ADD     #2,R2           ;UPDATE TO NEXT ADDRESS
2723  012042  005305                    DEC     R5              ;DONE A PASS?
2724  012044  001411                    BEQ     8$              ;IF YES BRANCH
2725  012046  011201           7$:     MOV     (R2),R1         ;GET DATA WRITTEN
2726  012050  020001                    CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2727  012052  001405                    BEQ     69$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2728  012054  004767  004544     68$:   JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2729  012060  004767  005602           JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2730  012064  000010                    .WORD   10              ;ERROR TYPE CODE.
2731  012066                     69$:
2732  012066  000732                    BR      4$
2733  012070  016705  167676     8$:    MOV     TEMP1,R5        ;RESTORE TEST WORD COUNTER FOR NEXT PASS.
2734  012074  005203                    INC     R3              ;INCREMENT      PASS COUNTER
2735  012076  000300                    SWAB    R0              ;SET UP NEW READ WORD
2736  012100  020327  000002           CMP     R3,#2           ;ADDRESSING UP?
2737  012104  001313                    BNE     3$              ;IF NO GO TO DOWN SEQUENCE
2738  012106  016702  167656           MOV     TEMP,R2         ;IF YES RESET ADDRESS TO START
2739  012112  000755                    BR      7$              ;GO TO UP SEQUENCE
2740  012114  062702  040000     9$:    ADD     #40000,R2       ;TESTING ENDS WITH R2 = BOTTOM OF BANK
2741                                                            ;ADD 8K FOR TEST START OF NEXT BANK
2742  012120  042702  001000           BIC     #1000,R2        ;CLEAR POSSIBLE FIRST STARTING
2743                                                            ;ADDRESS OF 1000 WHEN VECTOR AREA IS
2744                                                            ;PROTECTED.
2745  012124  004767  001062           JSR     PC,MMUP         ;UPDATE TO NEW BANK IF EXISTS
2746                                                            ;RETURN TO 100$
2747
2748                               ;;*****************************************************************
2749                               ;*TEST 24       WRITE CHECKERBOARD STARTING WITH '125252' DATA.
2750                               ;* THESE TESTS WRITE A CHECKERBOARD THROUGHOUT MEMORY,STALL
2751                               ;* FOR 2 SECONDS THEN CHECK PATTERN TO VERIFY DATA DID NOT
2752                               ;* DETERIORATE BETWEEN REFRESH CYCLES.
2753                               ;*
2754                               ;*      R0=DATA WRITTEN INTO MEMORY(SHOULD BE)
2755                               ;*      R1=DATA READ FROM MEMORY(WAS)
2756                               ;*      R2=VIRTUAL ADDRESS
2757                               ;*      R3=SMALL LOOP COUNTER FOR STALL
2758                               ;*      R4=NUMBER OF TIMES SMALL LOOP DONE
2759                               ;*      R5=BLOCK BOUNDARY BIT MASK.
2760                               ;;*****************************************************************
2761  012130                       TST24:
2762  012130  004567  005220           JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
2763  012134  004467  000640           JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2764  012140  012700  125252           MOV     #125252 ,R0     ;SETUP DATA PATTERN
2765  012144  010022           1$:     MOV     R0      ,(R2)+  ;WRITE A WORD
2766  012146  005100                    COM     R0              ;COMPLEMENT DATA
2767  012150  077503                    SOB     R5,     1$      ;BRANCH IF MORE IN CURRENT BLOCK
2768  012152  004767  001034           JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
2769  012156  005003                    CLR     R3              ;SET UP COUNTER FOR STALL
2770  012160  012704  000046           MOV     #46,    R4      ;DO LOOP 46 TIMES OR 2 SEC. TOTAL.
```

```
2771  012164  005303                  2$:     DEC     R3
2772  012166  001376                          BNE     2$
2773  012170  005304                          DEC     R4
2774  012172  001374                          BNE     2$
2775  012174  004467  000600                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2776  012200  012700  125252                  MOV     #125252,R0      ;INIT DATA FOR CHECKING
2777  012204                          3$:
2778  012204  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2779  012206  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2780  012210  001405                          BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2781  012212  004767  004406          64$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2782  012216  004767  005444                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2783  012222  000006                          .WORD   6               ;ERROR TYPE CODE.
2784  012224                          65$:
2785  012224  005100                          COM     R0
2786  012226  077512                          SOB     R5,     3$      ;BRANCH IF MORE IN CURRENT BLOCK
2787  012230  004767  000756                  JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
2788                                  ;;*****************************************************************
2789                                  ;*TEST 25        WRITE CHECKERBOARD STARTING WITH 052525 DATA
2790                                  ;;*****************************************************************
2791  012234                          TST25:
2792  012234  004567  005114                  JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
2793  012240  004467  000534                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2794  012244  012700  052525                  MOV     #052525,R0      ;SETUP DATA PATTERN
2795  012250  010022                  1$:     MOV     R0,     (R2)+   ;WRITE A WORD
2796  012252  005100                          COM     R0
2797  012254  077503                          SOB     R5,     1$      ;BRANCH IF MORE IN CURRENT BLOCK
2798  012256  004767  000730                  JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
2799  012262  005003                          CLR     R3              ;SET COUNTER FOR LOOP
2800  012264  012704  000046                  MOV     #46.,   R4      ;DO LOOP 46 TIMES OR 2 SEC. TOTAL
2801  012270  005303                  2$:     DEC     R3
2802  012272  001376                          BNE     2$
2803  012274  005304                          DEC     R4
2804  012276  001374                          BNE     2$
2805  012300  004467  000474                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2806  012304  012700  052525                  MOV     #052525,R0      ;INIT PATTERN FOR CHECKING
2807  012310                          3$:
2808  012310  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2809  012312  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2810  012314  001405                          BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2811  012316  004767  004302          64$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2812  012322  004767  005340                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2813  012326  000006                          .WORD   6               ;ERROR TYPE CODE.
2814  012330                          65$:
2815  012330  005100                          COM     R0
2816  012332  077512                          SOB     R5,     3$      ;BRANCH IF MORE IN CURRENT BLOCK
2817  012334  004767  000652                  JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
```

```
                                        .SBTTL DONE:    RELOCATE PROGRAM AND REPEAT ALL TESTS.
012340                                  DONE:
012340  004567  005010                          JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
012344  005067  166610                  TST32:  CLR     $TIMES          ;RESET ITERATION COUNTER FOR RESTARTING TEST.
012350  105067  166516                          CLRB    $TSTNM          ;RESET TEST NUMBER.
012354  032777  000200  166546                  BIT     #SW07,  @SWR    ;CHECK FOR INHIBIT RELOCATION SWITCH.
012362  001037                                  BNE     $EOP            ;SKIP RELOCATION IF SWITCH SET.
012364  105767  166624                          TSTB    $ENV            ;CHECK FOR APT11
012370  001403                                  BEQ     1$              ;BRANCH IF NOT APT11
012372  005767  166604                          TST     $PASS           ;CHECK FOR  1ST PASS
012376  001026                                  BNE     6$              ;IF APT,DO NOT RELOCATE AFTER FIRST PASS
012400  026767  165436  165440          1$:     CMP     42,46           ;CHECK FOR ACT11
012406  001003                                  BNE     2$              ;BRANCH IF NOT ACT11
012410  005767  166566                          TST     $PASS           ;CHECK FOR FIRST PASS
012414  001417                                  BEQ     6$              ;IF ACT,DO NOT RELOCATE ON FIRST PASS
012416  032767  000001  166110          2$:     BIT     #BIT0,PRGMAP    ;CHECK IF PROGRAM IN FIRST 8K
012424  001404                                  BEQ     4$              ;IF NOT IN FIRST 8K,RELOC TOP TO BOTTOM
                                                                        ;MUST BE XXDP OR STANDALONE
012426  004767  002140                          JSR     PC,     RELTOP  ;RELOCATE PROGRAM TO TOP OF MEMORY.
012432  000167  174224          3$:     JMP     START1                  ;LOOP BACK AND RUN ALL TESTS AGAIN.

012436  004767  002444          4$:     JSR     PC,     RELO    ;RELOCATE PROGRAM BACK TO FIRST 8K.
012442  005767  165374                  TST     42              ;TEST FOR XXDP
012446  001402                          BEQ     6$              ;IF NOT RUNNING UNDER MON. DONT
012450  004767  002632          5$:     JSR     PC,     RESLDR  ;RESTORE LOADERS.
012454                          6$:
012454  004567  007114                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
012460  001171                          .WORD   $CRLF           ;ADDRESS OF MESSAGE TO BE TYPED
```

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

                                .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 START1

012462                          $EOP:
012462  000240                          NOP
012464  005067  166470                  CLR     $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
012470  005267  166506                  INC     $PASS           ;;INCREMENT THE PASS NUMBER
012474  042767  100000  166500          BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
012502  005327                          DEC     (PC)+           ;;LOOP?
012504  000001                  $EOPCT: .WORD   1
012506  003041                          BGT     $DOAGN          ;;YES
012510  012737                          MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
012512  000001                  $ENDCT: .WORD   1
012514  012504                          $EOPCT
012516  004567  007052                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
012522  012616                          .WORD   $ENDMG          ;ADDRESS OF MESSAGE TO BE TYPED
012524  016746  166452                  MOV     $PASS,-(SP)     ;;SAVE $PASS FOR TYPEOUT
                                ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPDS ROUTINE
                                ;*  WIHTOUT USING A ''TRAP'' INSTRUCTION AS CALLED FOR BY **SYSMAC**.
012530  106746                          MFPS    -(SP)           ;PUT THE PROCESSOR STATUS ON THE STACK
012532  105066  000001                  CLRB    1(SP)           ;HIGH BYTE CLEARED TO INSURE KERNEL MODE
                                                                ;ON PSW RETURN.
012536  004767  010026                  JSR     PC,     $TYPDS  ;GO TO THE SUBROUTINE
012542  004567  007026                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
012546  012633                          .WORD   $ENULL          ;ADDRESS OF MESSAGE TO BE TYPED
012550                          $GET42:

012550  016700  165266                  MOV     42,     R0              ;;GET MONITOR ADDRESS
012554  001416                          BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
012556  000005                          RESET                   ;;CLEAR THE WORLD
012560  004710                  $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
012562  000240                          NOP                     ;;SAVE ROOM
012564  000240                          NOP                     ;;FOR
012566  000240                          NOP                     ;;ACT11
012570  023737  000042  000046          CMP     @#42,@#46       ;;ARE WE UNDER ACT11 OR XXDP
012576  001405                          BEQ     $DOAGN          ;;IF ACT11 THEN RESTART
012600  105737  001214                  TSTB    @#$ENV          ;;CHECK FOR APT11
012604  001002                          BNE     $DOAGN          ;;IF APT11 THEN RESTART
012606  004767  002562                  JSR     PC,     SAVLDR  ;;IF XXDP FIRST SAVE MONITOR
012612                          $DOAGN:
012612  000167  174044                  JMP     START1          ;;RETURN*****
012616  005015  047105  020104  $ENDMG: .ASCIZ  <15><12>/END PASS #/
012624  040520  051523  021440
012632     000
012633     377     377     000  $ENULL: .BYTE   -1,-1,0         ;;NULL CHARACTER STRING
                                .SBTTL SUBROUTINE AND TRAP ROUTINE SECTION.
                                .SBTTL  MEMORY MANAGEMENT AND ADDRESSING SUBROUTINES.
                                ;;*****************************************************************
                                ;*  SET UP ALL THE MEM MGMT REGISTERS FOR NORMAL OPERATION.
                                ;*  THE PROGRAM IS POINTED TO BY PARS 0 AND 1.
```

```
                                ;*  THE MEMORY UNDER TEST IS POINTED TO BY PARS 2 AND 3.
                                ;*  THE DEVICE ADDRESS AREA IS POINTED TO BY PAR 7.
                                ;*  PARS 4, 5, AND 6 ARE UNUSED.
                                ;;*****************************************************************
012636                          MMINIT:
012636  012737  077406  172300          MOV     #200-1*400+UP+RW,@#KIPDR0       ;SET KIPDR0   RW UP 200 BLOCKS
012644  012737  077406  172302          MOV     #200-1*400+UP+RW,@#KIPDR1       ;SET KIPDR1   RW UP 200 BLOCKS
012652  012737  077406  172304          MOV     #200-1*400+UP+RW,@#KIPDR2       ;SET KIPDR2 - RW UP 200 BLOCKS
012660  012737  077406  172306          MOV     #200-1*400+UP+RW,@#KIPDR3       ;SET KIPDR3 = RW UP 200 BLOCKS
012666  005037  172310                  CLR     @#KIPDR4
012672  005037  172312                  CLR     @#KIPDR5
012676  005037  172314                  CLR     @#KIPDR6
012702  012737  077406  172316          MOV     #200-1*400+UP+RW,@#KIPDR7       ;SET KIPDR7 = RW UP 200 BLOCKS
012710  005037  172340                  CLR     @#KIPAR0        ;MAP PAR0 INTO BANK0
012714  012737  000200  172342          MOV     #200, @#KIPAR1  ;MAP PAR1 INTO BANK1
012722  005037  172344                  CLR     @#KIPAR2        ;MAP PAR2 INTO BANK0
012726  005037  172346                  CLR     @#KIPAR3
012732  005037  172350                  CLR     @#KIPAR4
012736  005037  172352                  CLR     @#KIPAR5
012742  005037  172354                  CLR     @#KIPAR6
012746  012737  177600  172356          MOV     #177600, @#KIPAR7       ;MAP PAR7 INTO I/O BANK
012754  005767  165556                  TST     MMAVA           ;IF 22 BIT ADR NOT AVAILABLE
012760  100003                          BPL     1$              ;  THEN BRANCH
012762  052737  000020  172516          BIS     #BIT4,@#SR3     ;  ELSE ENABLE 22 BIT ADDRESSING
012770  012737  000001  177572  1$:     MOV     #1,@#SR0        ;ENABLE MEMORY MANAGEMENT
012776  000207                          RTS     PC              ;RETURN


                                ;;*****************************************************************
                                ;*  MEMORY ADDRESS POINTER INITIALIZATION ROUTINES.
                                ;;*****************************************************************

013000                          INITMM:
013000  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
013002  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
013004  012767  100000  166766          MOV     #BIT15,PLUS1
013012  005002                          CLR     R2              ;INITIALIZE ADDRESS
013014  005767  165516                  TST     MMAVA
013020  001410                          BEQ     2$
013022  012700  000020                  MOV     #20,R0
013026  016701  166762                  MOV     .BITPT,R1
013032  005021                  1$:     CLR     (R1)+
013034  077002                          SOB     R0,1$
013036  005037  172344                  CLR     @#KIPAR2
013042  012767  013204  166704  2$:     MOV     #INITEX,MMORE
013050  066767  165456  166676          ADD     RELOCF,MMORE
013056  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
013060  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
013062  004767  000124                  JSR     PC,MMUP
013066  004767  004574                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
                                ;FATAL ERROR
013072  000007                          .WORD   7               ;ERROR TYPE CODE.
013074  000000                          HALT            ;FATAL ERROR'.!  NO MEM INDICATED IN MEMMAP ABOVE 8K


013076                          INITDN:
```

```
2958  013076  010046                        MOV     R0,-(SP)        ;;PUSH R0 ON STACK
2959  013100  010146                        MOV     R1,-(SP)        ;;PUSH R1 ON STACK
2960  013102  005002                        CLR     R2
2961  013104  005767  165426                TST     MMAVA
2962  013110  001414                        BEQ     2$
2963  013112  012700  000020                MOV     #20,R0
2964  013116  016701  166672                MOV     .BITPT,R1
2965  013122  005021                1$:     CLR     (R1)+
2966  013124  077002                        SOB     R0,1$
2967  013126  012767  000001  166644        MOV     #BIT0,PLUS1
2968  013134  005037  172344                CLR     @#KIPAR2
2969  013140  000403                        BR      3$
2970  013142  012767  000020  166544  2$:   MOV     #BIT4,BITPT
2971  013150  012767  013204  166576  3$:   MOV     #INITEX,MMORE
2972  013156  066767  165350  166570        ADD     RELOCF,MMORE
2973  013164  012601                        MOV     (SP)+,R1        ;;POP STACK INTO R1
2974  013166  012600                        MOV     (SP)+,R0        ;;POP STACK INTO R0
2975  013170  004767  000362                JSR     PC,MMDOWN
2976  013174  004767  004466                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2977                                ;FATAL ERROR
2978  013200  000007                        .WORD   7               ;ERROR TYPE CODE.
2979  013202  000000                        HALT                    ;FATAL ERROR!.!  NO MEM INDICATED IN MEMMAP ABOVE 8K.
2980  013204  010467  166544        INITEX: MOV     R4,MMORE
2981  013210  000204                        RTS     R4
2982
2983                                ;;*****************************************************************
2984                                ;*  COMMON UPWARDS ADDRESSING ROUTINE
2985                                ;*  FINDS NEXT EXISTING 8K BANK AND UPDATES POINTERS.
2986                                ;*  GOES TO ADDRESS IN "MMORE" IF MORE BANKS,
2987                                ;*  DOES STRAIGHT EXIT WHEN ALL MEMORY HAS BEEN DONE.
2988                                ;;*****************************************************************
2989
2990  013212  012705  020000        MMUP:   MOV     #20000,R5       ;SET UP 8K TEST COUNTER
2991                                ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $CKSWR ROUTINE
2992                                ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
2993  013216  106746                        MFPS    -(SP)           ;PUT THE PROCESSOR STATUS ON THE STACK
2994  013220  105066  000001                CLRB    1(SP)           ;HIGH BYTE CLEARED TO INSURE KERNEL MODE
2995                                                                  ;ON PSW RETURN.
2996  013224  004767  005200                JSR     PC,     $CKSWR  ;GO TO THE SUBROUTINE
2997  013230  005767  165302                TST     MMAVA           ;CHECK FOR MEMORY MANAGEMENT
2998  013234  001475                        BEQ     7$              ;IF MMAVA = 0 ,BRANCH TO NON KT MODE
2999  013236  010046                        MOV     R0,-(SP)        ;;PUSH R0 ON STACK
3000  013240  010146                        MOV     R1,-(SP)        ;;PUSH R1 ON STACK
3001  013242  010346                        MOV     R3,-(SP)        ;;PUSH R3 ON STACK,
3002  013244  010446                        MOV     R4,-(SP)        ;;PUSH R4 ON STACK
3003  013246  012702  040000                MOV     #40000,R2       ;MMU AVAIL. SET UP VA   0,PAR2
3004  013252  005767  166522                TST     PLUS1           ;CHECK IF FIRST TIME ENTRY
3005  013256  001003                        BNE     100$            ;DO NOT ADD 8K TO PAR2
3006                                                                  ;UPON FIRST ENTRY
3007  013260  062737  000400  172344  1$:   ADD     #400,@#KIPAR2   ;ADD 8K TO PAR2
3008  013266  016701  166522        100$:   MOV     .BITPT,R1
3009  013272  012700  000020                MOV     #20,R0
3010  013276  006367  166476                ASL     PLUS1
3011  013302  006121                2$:     ROL     (R1)+           ;ROTATE TOTAL TABLE TO INSURE
3012  013304  077002                        SOB     R0,2$           ;ROTATION OF ONE BIT
3013  013306  103005                        BCC     101$            ;TESTING IS DONE WHEN CARRY BIT IS SET
```

```
3014  013310  012604                        MOV     (SP)+,R4        ;;POP STACK INTO R4
3015  013312  012603                        MOV     (SP)+,R3        ;;POP STACK INTO R3
3016  013314  012601                        MOV     (SP)+,R1        ;;POP STACK INTO R1
3017  013316  012600                        MOV     (SP)+,R0        ;;POP STACK INTO R0
3018  013320  000510                        BR      12$             ;DONE,READY TO EXIT
3019  013322  016703  166466        101$:   MOV     .BITPT,R3
3020  013326  016704  166456                MOV     .TSTMAP,R4
3021  013332  012700  000020                MOV     #20,R0
3022  013336  032324                3$:     BIT     (R3)+,(R4)+
3023  013340  001002                        BNE     4$
3024  013342  077003                        SOB     R0,3$
3025  013344  000745                        BR      1$
3026  013346  013737  172344  172346  4$:   MOV     @#KIPAR2,@#KIPAR3
3027  013354  062737  000200  172346        ADD     #200,@#KIPAR3
3028  013362  012604                        MOV     (SP)+,R4        ;;POP STACK INTO R4
3029  013364  012603                        MOV     (SP)+,R3        ;;POP STACK INTO R3
3030  013366  012601                        MOV     (SP)+,R1        ;;POP STACK INTO R1
3031  013370  012600                        MOV     (SP)+,R0        ;;POP STACK INTO R0
3032  013372  005767  165140                TST     MMAVA
3033  013376  100005                        BPL     5$
3034  013400  032767  100000  166344        BIT     #BIT15,BITPT+36
3035  013406  001005                        BNE     6$
3036  013410  000436                        BR      9$
3037  013412  032767  100000  166274  5$:   BIT     #BIT15,BITPT
3038  013420  001432                        BEQ     9$
3039  013422  012705  010000          6$:   MOV     #10000,R5
3040  013426  000427                        BR      9$
3041  013430  006367  166344          7$:   ASL     PLUS1
3042  013434  006167  166254                ROL     BITPT
3043  013440  103440                        BCS     12$
3044  013442  036767  166246  166104        BIT     BITPT,TSTMAP
3045  013450  001003                        BNE     8$
3046  013452  062702  040000                ADD     #40000,R2
3047  013456  000764                        BR      7$
3048  013460  032767  000010  166226  8$:   BIT     #BIT3,BITPT
3049  013466  001407                        BEQ     9$
3050  013470  012705  010000                MOV     #10000,R5
3051  013474  005767  166274                TST     FLG30K
3052  013500  001402                        BEQ     9$
3053  013502  062705  004000                ADD     #4000,R5
3054  013506  032767  000001  166200  9$:   BIT     #BIT0,BITPT
3055  013514  001407                        BEQ     10$
3056  013516  066702  166240                ADD     FSTADR,R2
3057  013522  005767  166234                TST     FSTADR          ;FSTADR = STARTING ADDRESS
3058                                                                ;     0 OR 1000 IF VECTOR AREA IS
3059                                                                ;     PROTECTED.
3060  013526  001402                        BEQ     10$
3061  013530  162705  000400                SUB     #400,R5         ;ADJUST ADDR COUNTER TO PROTECT 0-1000
3062  013534  016716  166214        10$:    MOV     MMORE,(SP)
3063  013540  000207                11$:    RTS     PC
3064                                  ;*  BEFORE FINAL EXIT, CHECK FOR ANY NON-TRAP PARITY ERRORS.
3065  013542  005767  167627        12$:    TST     MPRX            ;CHECK FOR ANY PARITY REGISTERS PRESENT.
3066  013546  001402                        BEQ     13$             ;BR IF NONE.
3067  013550  004767  002106                JSR     PC,CKPMER       ;CHECK FOR PARITY MEMORY ERRORS.
3068  013554  000207                13$:    RTS     PC              ;STRAIGHT RETURN.
3069
```

```
3070                                    ;;*****************************************************************
3071                                    ;*  MEMORY DOWNWARDS ADDRESSING SUBROUTINE.
3072                                    ;*  FINDS NEXT LOWER 8K BANK AND UPDATES POINTERS.
3073                                    ;*  GOES TO ADDRESS IN "MMORE" IF MORE BANKS,
3074                                    ;*  DOES STRAIGHT EXIT WHEN ALL MEMORY HAS BEEN DONE.
3075                                    ;;*****************************************************************
3076
3077  013556  012705  020000           MMDOWN: MOV     #20000,R5       ;SET UP 8K TEST COUNTER
3078                                    ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $CKSWR ROUTINE
3079                                    ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
3080  013562  106746                          MFPS    -(SP)           ;PUT THE PROCESSOR STATUS ON THE STACK
3081  013564  105066  000001                 CLRB    1(SP)           ;HIGH BYTE CLEARED TO INSURE KERNEL MODE
3082                                                                   ;ON PSW RETURN.
3083  013570  004767  004634                 JSR     PC,     $CKSWR  ;GO TO THE SUBROUTINE
3084  013574  005767  164736                 TST     MMAVA           ;CHECK FOR MEMORY MANAGEMENT
3085  013600  001502                          BEQ     7$              ;IF MMAVA = 0 THEN BR TO NON KT MODE
3086  013602  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
3087  013604  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
3088  013606  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
3089  013610  010446                          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
3090  013612  162737  000400  172344  1$:     SUB     #400,@#KIPAR2   ;DOWN DATE TO NEXT 8K BANK
3091  013620  016700  166170                 MOV     .BITPT,R0
3092  013624  062700  000040                 ADD     #40,R0          ;SET UP TO MOVE POINTER
3093  013630  012703  000020                 MOV     #20,R3          ;ONE POSITION THRU 20 WORD TABLE
3094  013634  006267  166140                 ASR     PLUS1           ;GET INITIAL VALUE OF BIT
3095  013640  006040                  2$:     ROR     -(R0)           ;MOVE THE POINTER
3096  013642  077302                          SOB     R3,2$
3097  013644  103005                          BCC     100$            ;CONTINUE TESTING UNTIL CARRY BIT SET
3098  013646  012604                          MOV     (SP)+,R4        ;;POP STACK INTO R4
3099  013650  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
3100  013652  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
3101  013654  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
3102  013656  000517                          BR      11$             ;TESTING DONE,READY TO EXIT
3103  013660  016700  166130          100$:   MOV     .BITPT,R0       ;THEN WE HAVE TESTED LAST BANK
3104  013664  062700  000040                 ADD     #40,R0          ;IF NO UNDERFLOW SET UP TO
3105  013670  016701  166114                 MOV     .TSTMAP,R1      ;SEE IF BANK SHOULD BE TESTED
3106  013674  062701  000040                 ADD     #40,R1
3107  013700  012703  000020                 MOV     #20,R3
3108  013704  034041                  3$:     BIT     -(R0),-(R1)     ;CHECK IF BANK IS UNDER TEST
3109  013706  001002                          BNE     4$              ;IF YES GO SET UP PARS
3110  013710  077303                          SOB     R3,3$           ;IF NO CHECK ALL POSSIBLES
3111  013712  000737                          BR      1$              ;IF NOT GO CHECK NEXT LOWER 8K
3112  013714  013737  172344  172346  4$:     MOV     @#KIPAR2,@#KIPAR3;MAKE PAR2 AND PAR3 POINT TO SAME
3113  013722  062737  000200  172346         ADD     #200,@#KIPAR3   ;MAKE KIPAR3 UPPER 4K
3114  013730  012702  100000                 MOV     #100000,R2      ;SET UP FIRST ADDRESS + 2
3115  013734  012604                          MOV     (SP)+,R4        ;;POP STACK INTO R4
3116  013736  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
3117  013740  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
3118  013742  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
3119  013744  005767  164566                 TST     MMAVA           ;CHECK FOR 22 BIT ADDRESSING
3120  013750  100005                          BPL     5$              ;IF NO,CHECK 18 BIT MODE
3121  013752  032767  100000  165772         BIT     #BIT15,BITPT+36 ;TESTING HIGHEST 22 BIT SEG.?
3122  013760  001005                          BNE     6$              ;IF YES,DO ONLY 4K SEGMENT
3123  013762  000442                          BR      9$              ;IF NO,DO 8K SEGMENT
3124  013764  032767  100000  165722  5$:     BIT     #BIT15,BITPT    ;TESTING HIGHESTT 18 BIT SEGMENT
3125  013772  001436                          BEQ     9$              ;IF NO ,DO 8K SEGMENT
```

```
3126  013774  012702  060000          6$:   MOV   #60000,R2       ;DO ONLY 4K SEGMENT
3127  014000  012705  010000                MOV   #10000,R5       ;SET UP 4K WORD COUNTER
3128  014004  000431                       BR    9$              ;GET OVER NON KT ROUTINE
3129  014006  006267  165702          7$:   ASR   BITPT           ;POINT TO NEXT LOWEST BANK
3130  014012  103441                       BCS   11$             ;IF CARRY FOR LSB THEN DONE TESTING
3131  014014  036767  165674  165532       BIT   BITPT,TSTMAP    ;IS THIS BANK UNDER TEST?
3132  014022  001003                       BNE   8$
3133  014024  162702  040000               SUB   #40000,R2
3134  014030  000766                       BR    7$
3135  014032  032767  000010  165654  8$:  BIT   #BIT3,BITPT
3136  014040  001413                       BEQ   9$
3137  014042  162702  020000               SUB   #20000,R2
3138  014046  012705  010000               MOV   #10000,R5
3139  014052  005767  165716               TST   FLG30K
3140  014056  001404                       BEQ   9$
3141  014060  062702  010000               ADD   #10000,R2
3142  014064  062705  004000               ADD   #4000,R5
3143  014070  032767  000001  165616  9$:  BIT   #BIT0,BITPT
3144  014076  001405                       BEQ   10$
3145  014100  005767  165656               TST   FSTADR          ;CHECK FIRST ADDRESS = 0 OR 1000
3146  014104  001402                       BEQ   10$
3147  014106  162705  000400               SUB   #400,R5         ;PROTECT VECTOR AREA 0-1000 FROM
3148                                                             ;BEING TESTED IF FSTADR =1000
3149  014112  016716  165636          10$:  MOV   MMORE,(SP)      ;SETUP RETURN FOR MORE TESTING
3150  014116  000207                11$:  RTS   PC                ;RETURN
3151
```

```
 3152                                   .SBTTL  SUBROUTINES FOR ADDRESS AND WORSE CASE NOISE TESTS.
 3153                                   ;;*****************************************************************
 3154                                   ;*  SUBROUTINE TO CALCULATE PHYSICAL ADDRESS AND PUT IT IN R0 (BOTTOM 16 BITS).
 3155                                   ;;*****************************************************************
 3156  014120  005767  164412          PHYADR: TST     MMAVA           ;CHECK FOR MEM MGMT AVAILABLE
 3157  014124  001433                          BEQ     4$              ;BRANCH IF NO MEM MGMT
 3158  014126  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
 3159  014130  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
 3160  014132  010446                          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
 3161  014134  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
 3162  014136  010005                  1$:     MOV     R0,R5           ;SAVE FOR PHYSICAL ADDR. ADDITION
 3163  014140  042700  017777                  BIC     #17777,R0       ;CLEAR VIRTUAL ADDR BITS
 3164  014144  012701  000014                  MOV     #12.,R1         ;SET UP ROTATE COUNTER
 3165  014150  006200                  2$:     ASR     R0              ;TO SHIFT PAR POINTER
 3166  014152  077102                          SOB     R1,2$           ;INTO BITS 0-3
 3167  014154  012704  172340                  MOV     #KISAR0,R4      ;STORE START OF PARS
 3168  014160  060004                          ADD     R0,R4           ;USE PAR POINTER IN 0-3 AS INDEX
 3169  014162  011403                          MOV     (R4),R3         ;STORE CONTENTS OF PROPER PAR
 3170  014164  012701  000006                  MOV     #6,R1           ;SET UP COUNTER TO
 3171  014170  006303                  3$:     ASL     R3              ;CONVERT PAR CONTENTS
 3172  014172  077102                          SOB     R1,3$           ;TO A PHYSICAL '%' ADDRESS
 3173  014174  010500                          MOV     R5,R0           ;STORE VIRTUAL ADDRESS
 3174  014176  042700  160000                  BIC     #160000,R0      ;CLEAR PAR POINTER
 3175  014202  060300                          ADD     R3,R0           ;ADD PHYSICAL '%' ADDRESS
 3176                                                                  ;TO OBTAIN 16 BIT PHYSICAL ADDRESS
 3177  014204  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
 3178  014206  012604                          MOV     (SP)+,R4        ;;POP STACK INTO R4
 3179  014210  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
 3180  014212  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
 3181  014214  000207                  4$:     RTS     PC              ;EXIT WITH PHYSICAL ADDR IN R0
 3182
 3183                                   ;;*****************************************************************
 3184                                   ;*  SUBROUTINE TO PUT BANK NUMBER INTO R0.
 3185                                   ;;*****************************************************************
 3186  014216  005000                  BANKNO: CLR     R0              ;INIT R0
 3187  014220  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
 3188  014222  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
 3189  014224  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
 3190  014226  010446                          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
 3191  014230  016701  165560                  MOV     .BITPT, R1      ;START OF BIT POINTER TABLE
 3192  014234  012703  000020                  MOV     #20,    R3      ;SET UP BITPT TABLE COUNTER
 3193  014240  012702  000020          1$:     MOV     #20,    R2      ;SET UP ROTATE COUNTER FOR 1 WORD
 3194  014244  012104                          MOV     (R1)+,R4        ;GET WORD TO ROTATE BIT
 3195  014246  006204                  2$:     ASR     R4              ;ROTATE BITPT WORD TO FIND BANK
 3196  014250  103410                          BCS     4$              ;IF CARRY SET ,THEN BANK FOUND
 3197  014252  105200                          INCB    R0              ;COUNT BANKS.
 3198  014254  001004                          BNE     3$              ;BR IF NOT OVERFLOW.
 3199  014256  004767  003404                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
 3200                                  ;FATAL ERROR
 3201  014262  000007                          .WORD   7               ;ERROR TYPE CODE.
 3202  014264  000000                          HALT                    ;FATAL ERROR!!!  NO POINTER FOUND.
 3203  014266  077211                  3$:     SOB     R2,     2$      ;IF NOT FINISHED ROTATING WORD
 3204                                                                  ;GO BACK AND ROTATE AGAIN
 3205  014270  077315                          SOB     R3,     1$      ;IF NOT FINISHED BITPT TABLE
 3206                                                                  ;THEN ROTATE THROUGH NEXT WORD.
 3207  014272                          4$:
```

```
3208  014272  012604                  MOV     (SP)+,R4        ;;POP STACK INTO R4
3209  014274  012603                  MOV     (SP)+,R3        ;;POP STACK INTO R3
3210  014276  012602                  MOV     (SP)+,R2        ;;POP STACK INTO R2
3211  014300  012601                  MOV     (SP)+,R1        ;;POP STACK INTO R1
3212  014302  000207                  RTS     PC              ;RETURN
3213
3214                            ;;************************************ ************************
3215                            ;*  SUBROUTINE TO WRITE THE CONSTANT IN R0 INTO ALL OF MEMORY.
3216                            ;;*************************************************************
3217  014304                    SETCON:
3218  014304  004467  176470          JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
3219  014310  010022           1$:    MOV     R0,     (R2)+   ;MOV CONSTANT INTO MEMORY
3220  014312  077502                  SOB     R5,     1$      ;BRANCH IF MORE IN CURRENT BLOCK
3221  014314  004767  176672          JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
3222  014320  000207                  RTS     PC              ;RETURN
3223
3224                            ;;*************************************************************
3225                            ;* ROUTINE TO ROTATE 'C' BIT THROUGH A MEMORY LOCATION.
3226                            ;;*************************************************************
3227  014322  106112            ROTATE: ROLB    (R2)            ;(R2)=177776 OR 000001
3228  014324  106112                    ROLB    (R2)            ;(R2)=177775 OR 000002
3229  014326  106112                    ROLB    (R2)            ;(R2)=177773 OR 000004
3230  014330  106112                    ROLB    (R2)            ;(R2)=177767 OR 000010
3231  014332  106112                    ROLB    (R2)            ;(R2)=177757 OR 000020
3232  014334  106112                    ROLB    (R2)            ;(R2)=177737 OR 000040
3233  014336  106112                    ROLB    (R2)            ;(R2)=177677 OR 000100
3234  014340  106112                    ROLB    (R2)            ;(R2)=177777 OR 000000
3235  014342  106122                    ROLB    (R2)+           ;(R2)=177577 OR 000200
3236  014344  106112                    ROLB    (R2)            ;(R2)=177377 OR 000400
3237  014346  106112                    ROLB    (R2)            ;(R2)=176777 OR 001000
3238  014350  106112                    ROLB    (R2)            ;(R2)=175777 OR 002000
3239  014352  106112                    ROLB    (R2)            ;(R2)=173777 OR 004000
3240  014354  106112                    ROLB    (R2)            ;(R2)=167777 OR 010000
3241  014356  106112                    ROLB    (R2)            ;(R2)=157777 OR 020000
3242  014360  106112                    ROLB    (R2)            ;(R2)=137777 OR 040000
3243  014362  106112                    ROLB    (R2)            ;(R2)=077777 OR 100000
3244  014364  106122                    ROLB    (R2)+           ;(R2)=177777 OR 000000
3245  014366  000207                    RTS     PC              ;RETURN
3246
3247                            ;;*************************************************************
3248                            ;*  SUBROUTINE TO WRITE 3 XOR 9 PATTERN INTO 256. WORD BLOCK.
3249                            ;;*************************************************************
3250  014370  012704  000020    W3X9:   MOV     #16.,R4         ;EACH LOOP WRITES 256. WORDS
3251
3252  014374  010022            2$:     MOV     R0,(R2)+
3253  014376  010022                    MOV     R0,(R2)+
3254  014400  010022                    MOV     R0,(R2)+
3255  014402  010022                    MOV     R0,(R2)+
3256
3257  014404  010322                    MOV     R3,(R2)+
3258  014406  010322                    MOV     R3,(R2)+
3259  014410  010322                    MOV     R3,(R2)+
3260  014412  010322                    MOV     R3,(R2)+
3261
3262  014414  010022                    MOV     R0,(R2)+
3263  014416  010022                    MOV     R0,(R2)+
```

```
3264  014420  010022                  MOV     R0,(R2)+
3265  014422  010022                  MOV     R0,(R2)+
3266
3267  014424  010322                  MOV     R3,(R2)+
3268  014426  010322                  MOV     R3,(R2)+
3269  014430  010322                  MOV     R3,(R2)+
3270  014432  010322                  MOV     R3,(R2)+
3271
3272  014434  005304                  DEC     R4
3273  014436  001356                  BNE     2$
3274  014440  010046                  MOV     R0,     -(SP)   ;SAVE R0
3275  014442  010300                  MOV     R3,     R0      ;PUT R3 INTO R0
3276  014444  012603                  MOV     (SP)+,  R3      ;PUT SAVED R0 INTO R3
3277  014446  000207                  RTS     PC              ;RETURN
```

```
                                .SBTTL  RELOCATION SUBROUTINES.
                                ;;*****************************************************************
                                ;* ROUTINE TO RELOCATE PROGRAM CODE
                                ;;*****************************************************************
014450                          RELOC:
014450  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
014452  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
014454  010446                          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
014456  012502                  4$:     MOV     (R5)+,  R2      ;GET FIRST LOCATION.
014460  012503                          MOV     (R5)+,  R3      ;GET FIRST LOCATION OF DESTINATION.
014462  012704  020000                  MOV     #20000, R4      ;SET UP 8K COUNTER.
014466  012223                  1$:     MOV     (R2)+,  (R3)+   ;MOV THE DATA.
014470  005304                          DEC     R4              ;COUNT THE WORDS.
014472  001375                          BNE     1$              ;BR IF MORE.
014474  012704  020000                  MOV     #20000, R4      ;RESET THE COUNTER.
014500  024243                  2$:     CMP     -(R2),  -(R3)   ;CHECK THE DATA JUST MOVED.
014502  001417                          BEQ     3$              ;BR IF DATA OK.
014504  011267  164404                  MOV     (R2),   $GDDAT  ;GET SOURCE DATA.
014510  011367  164402                  MOV     (R3),   $BDDAT  ;GET DESTINATION DATA.
014514  010267  164370                  MOV     R2,     $GDADR  ;GET SOURCE ADDRESS.
014520  010367  164366                  MOV     R3,     $BDADR  ;GET DESTINATION ADDRESS.
014524  004767  003136                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
014530  000023                          .WORD   23              ;ERROR TYPE CODE.
014532  000000                          HALT                    ;FATAL ERROR!!!  RELOCATION FAILED.
014534  162705  000004                  SUB     #4,     R5      ;ADJUST RETURN POINTER.
014540  000746                          BR      4$              ;GO BACK AND TRY AGAIN.
014542  005304                  3$:     DEC     R4              ;COUNT WORDS.
014544  001355                          BNE     2$              ;BR IF MORE.
014546  004567  005022                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
014552  024526                          .WORD   PRELOC          ;ADDRESS OF MESSAGE TO BE TYPED
                                                                ;'PROGRAM RELOCATED TO ''
014554  010346                          MOV     R3,     -(SP)   ;PUT THE DATA ON THE STACK.
014556  004767  006526                  JSR     PC,     $TYPAD  ;DETERMINE THE PHYSICAL ADDRESS AND TYPE IT.
014562  012604                          MOV     (SP)+,R4        ;;POP STACK INTO R4
014564  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
014566  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
014570  000205                          RTS     R5              ;RETURN
                                ;;*****************************************************************
                                ;*  SUBROUTINE TO MOVE PROGRAM FROM BOTTOM TO TOP 8K BANK OF 128KW  MEMORY.
                                ;*      HIGHEST AVAILABLE BANK WITH KT WILL BE 700000 - 737776
                                ;*      HIGHEST AVAILABLE BANK NON-KT  WILL BE 100000 - 137776
                                ;;*****************************************************************
014572  022767  000001  163734  RELTOP: CMP     #1,     PRGMAP  ;CHECK THAT THE PROGRAM IS NOW IN BANKS 0 AND 1.
014600  001404                          BEQ     1$              ;BR IF OK
014602  004767  003060                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
                                ;FATAL ERROR
014606  000007                          .WORD   7               ;ERROR TYPE CODE.
014610  000000                          HALT                    ;FATAL ERROR!!!  PROG SHOULD BE IN BANKS 0 AND 1
014612                          1$:
014612  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
014614  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
014616  005767  163714                  TST     MMAVA
014622  001441                          BEQ     10$
014624  012737  007600  172346          MOV     #7600,  @#KIPAR3 ;SET PAR TO TOP OF MEM
014632  012700  100000                  MOV     #BIT15, R0      ;...HI 128K.
014636  162737  000400  172346  2$:     SUB     #400,   @#KIPAR3        ;BACK DOWN ONE BANK.
```

```
3334  014644  006000                        ROR     R0              ;..LO 64K.
3335  014646  103445                        BCS     90$
3336  014650  030067  164640                BIT     R0,     MEMMAP  ;CHECK FOR BANK EXISTS.
3337  014654  001770                        BEQ     2$              ;BR IF NO BANK FOUND.
3338  014656  013737  172346  172344  3$:   MOV     @#KIPAR3,@#KIPAR2 ;COPY PAR
3339  014664  162737  000200  172344  4$:   SUB     #200,   @#KIPAR2 ;BACK DOWN WITH LOW PAR.
3340  014672  030067  163636                BIT     R0,     PRGMAP  ;CHECK FOR CONFLICT.
3341  014676  001031                        BNE     90$             ;ABORT IF DESTINATION OVERLAYS SOURCE.
3342  014700  004567  177544                JSR     R5,     RELOC   ;GO RELOCATE PROGRAM.
3343  014704  000000                        .WORD   0               ;SOURCE FIRST ADDRESS.
3344  014706  040000                        .WORD   40000           ;DESTINATION FIRST ADDRESS.
3345  014710  013737  172344  172340        MOV     @#KIPAR2,@#KIPAR0 ;RELOCATE LO BANK
3346  014716  013737  172346  172342        MOV     @#KIPAR3,@#KIPAR1 ;RELOCATE HI BANK.
3347                                ;*  PROGRAM SHOULD NOW BE EXECUTING OUT OF LAST TWO BANKS OF MEMORY.
3348  014724  000461                        BR      30$             ;BR TO COMMON EXIT.
3349
3350  014726  012700  000010        10$:    MOV     #BIT3,  R0      ;SET BANK POINTER TO TOP OF MEM.
3351  014732  012701  140000                MOV     #140000,R1      ;SET ADDRESS POINTER TO TOP
3352  014736  162701  040000        11$:    SUB     #40000, R1      ;BACK DOWN ONE BANK.
3353  014742  006200                        ASR     R0              ;MOVE POINTER DOWN ONE BANK.
3354  014744  103406                        BCS     90$             ;BR IF OVERFLOW.
3355  014746  030067  164542                BIT     R0,     MEMMAP  ;CHECK IF THIS BANK EXISTS.
3356  014752  001771                        BEQ     11$             ;BR IF NON-EXISTANT BANK.
3357  014754  030067  163554                BIT     R0,     PRGMAP  ;CHECK FOR A PROGRAM CONFLICT.
3358  014760  001404                        BEQ     12$             ;BR IF NO CONFLICT.
3359  014762                        90$:
3360  014762  004767  002700                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
3361                                ;FATAL ERROR
3362  014766  000007                        .WORD   7               ;ERROR TYPE CODE.
3363  014770  000000                        HALT                    ;FATAL ERROR!!!  NOT ENOUGH MEMORY??
3364  014772  010167  000006        12$:    MOV     R1,     13$     ;SET DATA FOR RELOCATION SUBROUTINE.
3365  014776  004567  177446                JSR     R5,     RELOC   ;GO RELOCATE THE PROGRAM TO TOP OF MEM.
3366  015002  000000                        .WORD   0               ;SOURCE STARTING ADDRESS.
3367  015004  000000                13$:    .WORD   0               ;DESTINATION STARTING ADDRESS.
3368  015006  010167  163520                MOV     R1,     RELOCF  ;SET RELOCATION FACTOR IN UNRELOCATED CODE.
3369  015012  060107                        ADD     R1,     PC      ;JUMP TO RELOCATED PROGRAM
3370                                ;*  PROGRAM NOW EXECUTING OUT OF TOP OF MEMORY.
3371  015014  060106                        ADD     R1,     SP      ;ADJUST THE STACK POINTER TO TOP OF MEMORY.
3372  015016  010167  163510                MOV     R1,     RELOCF  ;SET THE RELOCATION FACTOR.
3373  015022  060137  000004                ADD     R1,     @#ERRVEC ;ADJUST ERROR VECTOR.
3374  015026  060137  000024                ADD     R1,     @#PWRVEC ;ADJUST POWER FAIL VECTOR.
3375  015032  060137  000114                ADD     R1,     @#PARVEC ;ADJUST PARITY ERROR VECTOR.
3376  015036  060167  164066                ADD     R1,     SWR     ;ADJUST SOFTWARE SWITCH REGISTER.
3377  015042  060167  164064                ADD     R1,     DISPLAY ;ADJUST SOFTWARE DISPLAY REGISTER.
3378  015046  062701  002002        14$:    ADD     #RADTAB,R1      ;POINT TO THE RELATIVE RELOCATION TABLE.
3379  015052  066721  163454        15$:    ADD     RELOCF, (R1)+   ;ADD RELOCATION FACTOR TO ADDRESSES IN TABLE.
3380  015056  005721                16$:    TST     (R1)+           ;CHECK FOR INTERUM TERMINATOR.
3381  015060  001776                        BEQ     16$             ;BR SO AS TO NOT MODIFY ZERO.
3382  015062  024127  177777                CMP     -(R1),  #-1     ;CHECK FOR END OF TABLE.
3383  015066  001371                        BNE     15$             ;BR IF MORE IN TABLE.
3384  015070  010067  163440        30$:    MOV     R0,     PRGMAP  ;SET NEW PROGRAM MAP...LO 64K.
3385  015074  012601                        MOV     (SP)+,R1        ;;POP STACK INTO R1
3386  015076  012600                        MOV     (SP)+,R0        ;;POP STACK INTO R0
3387  015100  066716  163426                ADD     RELOCF, (SP)    ;ADJUST RETURN ADDRESS.
3388  015104  000207                        RTS     PC              ;RETURN
3389
```

```
3390                                    ;;*****************************************************************
3391                                    ;*  SUBROUTINE TO RELOCATE PROGRAM BACK TO BANKS 0 AND 1.
3392                                    ;;*****************************************************************
3393  015106  032767  000001  163420  RELO:   BIT     #1,     PRGMAP  ;CHECK FOR PROGRAM ALREADY IN BANKS 0 OR 1.
3394  015114  001404                          BEQ     1$              ;BR IF NO CONFLICT.
3395  015116  004767  002544                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
3396                                  ;FATAL ERROR
3397  015122  000007                          .WORD   7               ;ERROR TYPE CODE.
3398  015124  000000                          HALT                    ;FATAL ERROR!!!  PROGRAM ALREADY IN BANKS 0 OR 1.
3399  015126  005767  163404          1$:     TST     MMAVA           ;CHECK FOR MEM MGMT.
3400  015132  001417                          BEQ     10$             ;BR IF NO MEMMGMT.
3401  015134  005037  172344                  CLR     @#KIPAR2        ;SET PAR 2 TO BANK 0.
3402  015140  012737  000200  172346          MOV     #200,   @#KIPAR3 ;SET PAR 3 TO BANK 1.
3403  015146  004567  177276                  JSR     R5,     RELOC   ;GO MOVE 8K INTO BANKS 0 AND 1.
3404  015152  000000                          .WORD   0               ;SOURCE STARTING ADDRESS.
3405  015154  040000                          .WORD   40000           ;DESTINATION STARTING ADDRESS.
3406  015156  005037  172340                  CLR     @#KIPAR0        ;RESTORE PAR 0 TO BANK0.
3407  015162  012737  000200  172342          MOV     #200,   @#KIPAR1 ;RESTORE PAR 1 TO BANK 1.
3408                                  ;*  PROGRAM IS NOW EXECUTING OUT OF BANKS 0 AND 1.
3409  015170  000440                          BR      30$             ;BR TO COMMON EXIT.
3410
3411  015172  016746  163334          10$:    MOV     RELOCF, -(SP)   ;PUT RELOCATION FACTOR ONTO THE STACK.
3412  015176  011667  000004                  MOV     (SP),   20$     ;SET DATA FOR RELOC SUBROUTINE.
3413  015202  004567  177242                  JSR     R5,     RELOC   ;GO MOVE THE PROGRAM BACK TO BANKS 0 AND 1.
3414  015206  000000                  20$:    .WORD   0               ;SOURCE STARTING ADDRESS.
3415  015210  000000                          .WORD   0               ;DESTINATION STARTING ADDRESS.
3416  015212  161607                          SUB     (SP),   PC      ;JUMP TO RELOCATED PROGRAM.
3417                                  ;*  THE PROGRAM IS NOW EXECUTING OUT OF BANKS 0 AND 1.
3418  015214  161606                          SUB     (SP),   SP      ;RESET THE STACK POINTER.
3419  015216  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
3420  015220  012700  002002                  MOV     #RADTAB,R0      ;SET UP POINTER TO RELATIVE ADDRESS TABLE.
3421
3422  015224  166620  000002          21$:    SUB     2(SP),  (R0)+   ;RESET ADDRESSES TO UNRELOCATED VALUES.
3423  015230  005720                  22$:    TST     (R0)+           ;CHECK FOR TERMINATORS.
3424  015232  001776                          BEQ     22$             ;BR OVER TERMINATORS.
3425  015234  024027  177777                  CMP     -(R0),  #-1     ;CHECK FOR END OF TABLE INDICATOR.
3426  015240  001371                          BNE     21$             ;BR IF MORE ADDRESSES IN TABLE.
3427  015242  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
3428  015244  161637  000004                  SUB     (SP),   @#ERRVEC ;ADJUST ERROR VECTOR.
3429  015250  161637  000024                  SUB     (SP),   @#PWRVEC ;ADJUST POWER FAIL VECTOR.
3430  015254  161637  000114                  SUB     (SP),   @#PARVEC ;ADJUST PARITY ERROR VECTOR.
3431  015260  161667  163644                  SUB     (SP),   SWR     ;ADJUST SOFTWARE SWITCH REGISTER.
3432  015264  161667  163642                  SUB     (SP),   DISPLAY ;ADJUST SOFTWARE DISPLAY REGISTER.
3433  015270  162616                  23$:    SUB     (SP)+,  (SP)    ;ADJUST RETURN ADDRESS.
3434  015272  005067  163234          30$:    CLR     RELOCF          ;RESET RELOCATION FACTOR.
3435  015276  012767  000001  163230          MOV     #1,     PRGMAP  ;SET PROGRAM MAP TO POINT TO BANKS 0 AND 1.
3436  015304  000207                          RTS     PC              ;RETURN.
3437
3438                                    ;;*****************************************************************
3439                                    ;* THIS SUBROUTINE MOVES THE LOADER AREA BACK TO THE "TOP" OF MEMORY FROM
3440                                    ;*  WHENCE IT CAME.  THE LOADER AREA IS SAVED AT THE END OF THE 8K OF
3441                                    ;*  PROGRAM CODE WHEN THE PROGRAM IS INITIALLY RUN.
3442                                    ;;*****************************************************************
3443  015306  016700  164176  RESLDR: MOV     LMAD,   R0      ;CHECK IF THE LOADERS WERE SAVED.
3444  015312  001004                          BNE     1$              ;BR IF LOADER AREA WAS SAVED.
3445  015314  004767  002346                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
```

```
3446                                    ;FATAL ERROR
3447  015320  000007                            .WORD   7               ;ERROR TYPE CODE.
3448  015322  000000                            HALT                    ;FATAL ERROR!!!  CAN'T RESTORE LOADER AREA IF IT WASN'T SAVED.
3449  015324  005767  163206            1$:     TST     MMAVA           ;CHECK FOR MEM MGMT.
3450  015330  001402                            BEQ     2$              ;SKIP IF NO MEM MGMT.
3451  015332  005037  177572                    CLR     @#SR0           ;DISABLE MEM MGMT.
3452  015336  012701  040000            2$:     MOV     #40000, R1      ;GET END OF 8K, ASSUME PROG NOT RELOCATED.
3453  015342  012702  003000                    MOV     #1536., R2      ;GET COUNTER.
3454  015346  014140                    3$:     MOV     -(R1), -(R0)    ;MOVE THE LOADER AREA.
3455  015350  005302                            DEC     R2              ;COUNT HOW LONG THE AREA IS.
3456  015352  001375                            BNE     3$              ;BR IF NOT MORE TO MOVE.
3457  015354  005067  164130                    CLR     LMAD            ;CLEAR MONITOR SAVED FLAG
3458  015360  005767  163152                    TST     MMAVA           ;CHECK FOR MEM MGMT.
3459  015364  001402                            BEQ     4$              ;BR IF NO MEM MGMT.
3460  015366  005237  177572                    INC     @#SR0           ;ENABLE MEM MGMT.
3461  015372  000207                    4$:     RTS     PC              ;RETURN.
3462
3463                                    ;*  ROUTINE TO SAVE THE LOADERS AT THE END OF 8K.
3464  015374  005767  164110            SAVLDR: TST     LMAD            ;CHECK IF LOADERS HAVE BEEN SAVED ALREADY.
3465  015400  001024                            BNE     4$              ;BRANCH IF ALREADY SAVED
3466  015402  012700  040000                    MOV     #40000, R0      ;GET END OF 8K
3467  015406  010001                            MOV     R0,     R1      ;GET END OF 8K
3468  015410  012737  015422  000004            MOV     #2$, @#ERRVEC   ;SET UP TIMEOUT VECTOR
3469  015416  011020                    1$:     MOV     (R0),   (R0)+   ;SEARCH FOR END OF MEMORY
3470  015420  000776                            BR      1$              ;KEEP SEARCHING
3471  015422  022626                    2$:     CMP     (SP)+,  (SP)+   ;RESTORE STACK POINTER
3472  015424  012737  023252  000004            MOV     #ERRTRP,@#ERRVEC ;RESET TIMEOUT VECTOR.
3473  015432  010046                            MOV     R0,     -(SP)   ;SAVE LAST MEMORY ADDRESS (CONTIGUOUS)
3474  015434  012702  003000                    MOV     #1536., R2      ;SET UP WORD COUNTER
3475  015440  014041                    3$:     MOV     -(R0), -(R1)    ;SAVE THE LOADERS
3476  015442  005302                            DEC     R2              ;COUNT THE WORDS
3477  015444  001375                            BNE     3$              ;BRANCH IF MORE WORDS
3478  015446  012667  164036                    MOV     (SP)+,  LMAD    ;SAVE LAST MEMORY ADDRESS
3479  015452  000207                    4$:     RTS     PC              ;RETURN
```

```
                                .SBTTL  PARITY MEMORY TRAP SERVICE AND SUBROUTINES.
                                ;;*****************************************************************
                                ;* PARITY MEMORY UNEXPECTED ERROR TRAP SERVICE ROUTINE.
                                ;*  FIND OUT WHICH REGISTER DETECTED THE ERROR.
                                ;*  THEN SCAN MEMORY TO SEE IF PARITY ERROR STILL SET AND REPORT LOCATION.
                                ;;*****************************************************************
015454  011667  163432          PESRV:  MOV     (SP),   $BDADR  ;GET PC OF INSTRUCTION WHICH CAUSED ERROR.
015460  004567  004110                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
015464  024465                          .WORD   UNEXPT          ;ADDRESS OF MESSAGE TO BE TYPED
                                                                ;'UNEXPECTED MEMORY PARITY TRAP.'
015466  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
015470  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
015472  016703  164324                  MOV     .MPRX,  R3      ;GET POINTER TO PARITY REGISTERS.
015476  005733                  1$:     TST     @(R3)+          ;CHECK THE PARITY REG FOR AN ERROR FLAG.
015500  100415                          BMI     3$              ;BR IF THIS REGISTER SHOWS THE ERROR.
015502  005713                          TST     (R3)            ;CHECK FOR TABLE TERMINATOR.
015504  001374                          BNE     1$              ;BR IF MORE REGISTERS.
015506  004767  002154                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
                                ;***ERROR***    NO REGISTER INDICATED ERROR
015512  000024                          .WORD   24              ;ERROR TYPE CODE.
015514  000417                          BR      4$              ;EXIT
015516  005713                  2$:     TST     (R3)            ;CHECK FOR TABLE TERMINATOR.
015520  001415                          BEQ     4$              ;BR IF NO MORE PARITY REGISTERS.
015522  005733                          TST     @(R3)+          ;CHECK THE PARITY REG FOR AN ERROR FLAG.
015524  100374                          BPL     2$              ;BR IF NO ERROR FLAG.
015526  004567  004042                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
015532  024556                          .WORD   MTOE            ;ADDRESS OF MESSAGE TO BE TYPED
                                                                ;'MORE THAN ONE ERROR FOUND.'
015534                          3$:
015534  004767  001010          64$:    JSR     PC,     SPRNTQ  ;SET UP VALUES FOR ERROR PRINTING.
015540  004767  002122                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
015544  000025                          .WORD   25              ;ERROR TYPE CODE.
015546  004767  000262                  JSR     PC,     PSCAN   ;GO SCAN MEMORY FOR BAD PARITY.
015552  000761                          BR      2$              ;GO LOOK FOR MORE ERRORS.
015554                          4$:
015554  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
015556  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
015560  000002                          RTI                     ;RETURN.

                                ;;*****************************************************************
                                ;ROUTINE TO ENABLE PARITY ERROR ACTION ON MA/MF PARITY MEMORIES
                                ;THIS ROUTINE IS MEANT TO CATCH UNEXPECTEDS
                                ;;*****************************************************************
015562  005767  165602          MAMF:   TST     MPRX            ;CHECK IF ANY PARITY REGISTERS EXIST.
015566  001434                          BEQ     MAMF2           ;EXIT IF NO PARITY REGISTERS.
015570  032777  000100  163332          BIT     #SW6,   @SWR    ;CHECK FOR INHIBIT PARITY ERROR DETECTION.
015576  001030                          BNE     MAMF2           ;EXIT IF NO PARITY ERROR DETECTION.
015600  005767  162726                  TST     RELOCF          ;CHECK IF PROGRAM RELOCATED OUT OF BANK 0.
015604  001410                          BEQ     SETAE           ;BR IF PROG IN BANK 0.
015606  032777  000040  163314          BIT     #SW5,   @SWR    ;CHECK IF VECTORS PROTECTED.
015614  001004                          BNE     SETAE           ;BR IF VECTOR AREA PROTECTED.
015616  026727  164140  001000          CMP     FSTADR, #1000   ;CHECK FOR STARTING ADDRESS ABOVE THE VECTORS.
015624  103415                          BLO     MAMF2           ;EXIT IF VECTORS EXPOSED TO TESTING.

015626  016737  164176  000114  SETAE:  MOV     .PESRV, @#PARVEC ;SET PARITY ERROR TRAP VECTOR
015634  005037  000116                  CLR     @#PARVEC+2      ;PRIORITY LEVEL 0 ON TRAP
```

```
3536  015640  010346                        MOV     R3,-(SP)        ;;PUSH R3 ON STACK
3537  015642  016703  164154                MOV     .MPRX,  R3      ;GET PARITY REGISTER TABLE POINTER.
3538  015646  052733  000001        MAMF1:  BIS     #AE,    @(R3)+  ;SET ACTION ENABLE BIT IN PARITY REG
3539  015652  005713                        TST     (R3)            ;CHECK FOR END OF TABLE.
3540  015654  001374                        BNE     MAMF1           ;BR IF MORE PARITY REGISTERS.
3541  015656  012603                        MOV     (SP)+,R3        ;;POP STACK INTO R3
3542  015660  000207                MAMF2:  RTS     PC              ;RETURN.
3543
3544                                ;;*****************************************************************
3545                                ;*  SUBROUTINE TO CHECK PARITY REGISTERS FOR ERRORS THAT DIDN'T TRAP.
3546                                ;;*****************************************************************
3547  015662                        CHGG2:
3548  015662  005767  165502        CKPMER: TST     MPRX            ;CHECK IF ANY PARITY REGISTERS EXIST.
3549  015666  001437                        BEQ     4$              ;BR IF NO PARITY REGISTERS.
3550  015670  032777  000100  163232        BIT     #SW6,   @SWR    ;CHECK FOR INHIBIT PARITY ERROR CHECKING.
3551  015676  001033                        BNE     4$              ;BR IF PARITY ERROR CHECKING INHIBITED.
3552  015700  010346                        MOV     R3,-(SP)        ;;PUSH R3 ON STACK
3553  015702  016703  164114                MOV     .MPRX,  R3      ;GET PARITY REG TABLE POINTER.
3554  015706  005733                1$:     TST     @(R3)+          ;CHECK THE PARITY REG FOR AN ERROR FLAG.
3555  015710  100023                        BPL     3$              ;BR IF NO ERROR
3556  015712  032773  000001  177776        BIT     #BIT0,  @-2(R3) ;CHECK IF A TRAP SHOULD HAVE OCCURRED.
3557  015720  001410                        BEQ     2$              ;BR IF NO ACTION ENABLE.   CHGG2
3558  015722  004767  000622        64$:    JSR     PC,     SPRNTQ  ;SET UP VALUES FOR ERROR PRINTING.
3559  015726  004767  001734                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
3560  015732  000026                        .WORD   26              ;ERROR TYPE CODE.
3561  015734  000411                        BR      3$
3562  015736  004767  000072                JSR     PC,     PSCAN   ;GO SCAN ALL MEMORY FOR PARITY ERRORS.
3563  015742                        2$:
3564  015742  004767  000602        65$:    JSR     PC,     SPRNTQ  ;SET UP VALUES FOR ERROR PRINTING.
3565  015746  004767  001714                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
3566  015752  000027                        .WORD   27              ;ERROR TYPE CODE.
3567  015754  004767  000054                JSR     PC,     PSCAN   ;GO SCAN ALL MEMORY FOR PARITY ERRORS.
3568  015760  005713                3$:     TST     (R3)            ;CHECK FOR TABLE TERMINATOR.
3569  015762  001351                        BNE     1$              ;BR IF MORE.
3570  015764  012603                        MOV     (SP)+,R3        ;;POP STACK INTO R3
3571  015766  000207                4$:     RTS     PC              ;RETURN.
3572
3573                                ;;*****************************************************************
3574                                ;* SUBROUTINE TO VERIFY THAT THE PARITY CSR DOES CONTROL THE BANK
3575                                ;* UNDER TEST. MATCH THE BANK UNDER TEST WITH THE PARITY MAP TABLE.
3576                                ;* RETURN IF MEM BANK HAS BEEN MAPPED IN PARITY MAP TABLE.
3577                                ;* RETURN+2 IF ERROR,MEM BANK HAS NOT BEEN MAPPED.
3578                                ;;*****************************************************************
3579
3580  015770                        PARMAT:
3581  015770  010046                        MOV     R0,-(SP)        ;;PUSH R0 ON STACK
3582  015772  010146                        MOV     R1,-(SP)        ;;PUSH R1 ON STACK
3583  015774  010446                        MOV     R4,-(SP)        ;;PUSH R4 ON STACK
3584  015776  012700  000020                MOV     #20,R0          ;COUNT OF PARITY MAP TABLE
3585  016002  016701  164006                MOV     .BITPT,R1       ;MEM BANK UNDER TEST POINTER
3586  016006  010304                        MOV     R3,R4           ;STORE PARITY MAP TABLE
3587  016010  005724                        TST     (R4)+           ;MOVE PAST CSR ADDRESS TO MAP TABLE
3588  016012  032124                1$:     BIT     (R1)+,(R4)+     ;CHECK FOR TABLE MATCH
3589  016014  001003                        BNE     2$              ;BR IF MATCH MADE IN TABLES
3590  016016  005300                        DEC     R0              ;DO NEXT WORD IN TABLE
3591  016020  001374                        BNE     1$              ;BR TO NEXT WORD IN TABLE
```

```
3592  016022  005725                          TST     (R5)+           ;NO MATCH MADE,RETURN +2
3593  016024                          2$:
3594  016024  012604                          MOV     (SP)+,R4        ;;POP STACK INTO R4
3595  016026  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
3596  016030  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
3597  016032  000205                          RTS     R5
3598
3599
3600                                  ;;*****************************************************************
3601                                  ;* AFTER A PARITY ERROR IS ENCOUNTERED THIS SUBROUTINE IS USED
3602                                  ;* TO SCAN ALL PARITY MEMORY PER THE PMEMAP TABLE LOOKING FOR BYTE LOCATIONS
3603                                  ;* CAUSING THE PARITY ERROR.  UPON FINDING THE LOCATION(S) A MESSAGE
3604                                  ;* WILL BE TYPED AND THE LOCATION WILL BE REWRITTEN IN ORDER TO RESTORE GOOD PARITY.
3605                                  ;*      STORAGE USED:
3606                                  ;*              R0      = BANK POINTER
3607                                  ;*              R1      - PARITY CSR TABLE POINTER (MPRX)
3608                                  ;*              R2      - ADDRESS POINTER
3609                                  ;*              R3      - PMEMAP TABLE ENTRY
3610                                  ;*              R4      - ERROR DETECT FLAG
3611                                  ;*              TEMP    - TEMPORARY STORAGE
3612                                  ;*****************************************************************
3613  016034                          PSCAN:
3614  016034  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
3615  016036  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
3616  016040  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
3617  016042  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
3618  016044  010446                          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
3619  016046  004567  003522                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
3620  016052  024622                          .WORD   SCANM           ;ADDRESS OF MESSAGE TO BE TYPED
3621                                                                  ;''SCANNING MEMORY FOR BAD PARITY''
3622  016054  004767  000434                  JSR     PC,CLRPAR       ;GO INITIALIZE PARITY CSR'S
3623  016060  013746  000114                  MOV     @#114,-(SP)     ;;PUSH @#114 ON STACK
3624  016064  013746  000116                  MOV     @#116,-(SP)     ;;PUSH @#116 ON STACK
3625  016070  012737  000116  000114          MOV     #116,@#114      ;SET UP PARITY VECTOR TO
3626  016076  005037  000116                  CLR     @#116           ;HALT IF ANOTHER PARITY ERROR OCCURS
3627  016102  005004                          CLR     R4              ;INITIALIZE ERROR DETECT FLAG
3628  016104  005002                          CLR     R2              ;INITIALIZE ADDRESS POINTER
3629  016106  012700  000001                  MOV     #BIT0,R0        ;INITIALIZE BANK POINTER
3630  016112  016701  163674                  MOV     .PMEMAP,R1      ;POINT TO PMEMAP TABLE
3631  016116  005767  162414                  TST     MMAVA           ;IF MEMORY MANAGEMENT NOT AVAILABLE
3632  016122  001413                          BEQ     1$              ;  THEN BRANCH
3633  016124  013746  172344                  MOV     @#KIPAR2,-(SP)          ;;PUSH @#KIPAR2 ON STACK
3634  016130  013746  172346                  MOV     @#KIPAR3,-(SP)          ;;PUSH @#KIPAR3 ON STACK
3635  016134  005037  172344                  CLR     @#KIPAR2        ;INITIALIZE PAR2 TO 0
3636  016140  012737  000200  172346          MOV     #200,@#KIPAR3   ;INITIALIZE PAR3
3637  016146  012702  040000                  MOV     #40000,R2       ;INITIALIZE ADDRESS POINTER TO MAP THRU PAR2
3638
3639  016152  030011                  1$:     BIT     R0,(R1)         ;IF 8K BANK DOES EXIST
3640  016154  001003                          BNE     2$              ;  THEN BRANCH
3641  016156  062702  040000                  ADD     #40000,R2       ;  ELSE UPDATE ADDRESS TO NEXT 8K BANK
3642  016162  000475                          BR      10$             ;    AND BRANCH TO SEE IF SHOULD CONTINUE TESTING
3643
3644  016164  111267  163600          2$:     MOVB    (R2),TEMP       ;READ THE BYTE LOCATION
3645  016170  016703  163626                  MOV     .MPRX,R3        ;POINT TO PARITY CSR TABLE
3646
3647  016174  005733                  3$:     TST     @(R3)+          ;IF NO ERROR FLAG SET IN CSR
```

```
3648  016176  100024                        BPL     5$              ;  THEN BRANCH
3649  016200  005704                        TST     R4              ;  ELSE IF NOT FIRST ERROR
3650  016202  001003                        BNE     4$              ;     THEN BRANCH
3651  016204  005367  162672                DEC     $ERTTL          ;     ELSE ADJUST ERROR COUNT
3652  016210  005204                        INC     R4              ;       AND SET ERROR DETECT FLAG
3653
3654  016212                          4$:
3655  016212  004767  000332          64$:  JSR     PC,     SPRNTQ  ;SET UP VALUES FOR ERROR PRINTING.
3656  016216  004767  001444                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
3657  016222  000030                        .WORD   30              ;ERROR TYPE CODE.
3658  016224  111212                        MOVB    (R2),(R2)       ;REWRITE LOCATION TO CLEAR BAD PARITY
3659  016226  005053                        CLR     @-(R3)          ;CLEAR ERROR FLAG IN PARITY CSR
3660  016230  105712                        TSTB    (R2)            ;REREAD LOCATION TO SEE IF ERROR CLEARED
3661  016232  005733                        TST     @(R3)+          ;IF ERROR CLEARED
3662  016234  001405                        BEQ     5$              ;  THEN BRANCH
3663  016236  004567  003332                JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
3664  016242  024664                        .WORD   PEWNC           ;ADDRESS OF MESSAGE TO BE TYPED
3665                                                                ;'PARITY ERROR WILL NOT CLEAR''
3666  016244  005073  177776                CLR     @-2(R3)         ;  CLEAR PARITY CSR
3667
3668  016250  005713                  5$:   TST     (R3)            ;IF NOT DONE WITH TABLE
3669  016252  001350                        BNE     3$              ;  THEN TRY NEXT CSR FOR ERROR
3670  016254  005202                        INC     R2              ;  ELSE UPDATE ADDRESS POINTER
3671  016256  005767  162254                TST     MMAVA           ;IF MEMORY MANAGEMENT AVAILABLE
3672  016262  001013                        BNE     7$              ;  THEN BRANCH
3673  016264  005767  163504                TST     FLG30K          ;  ELSE IF NOT 30K SYSTEM
3674  016270  001404                        BEQ     6$              ;     THEN BRANCH
3675  016272  022702  170000                CMP     #170000,R2      ;     ELSE IF 30K I/O PAGE
3676  016276  001467                        BEQ     14$             ;        THEN ALL DONE CHECKING MEMORY
3677  016300  000423                        BR      9$              ;        ELSE TRY SOMEMORE
3678
3679  016302  022702  160000          6$:   CMP     #160000,R2      ;IF 28K I/O PAGE
3680  016306  001463                        BEQ     14$             ;  THEN ALL DONE CHECKING MEMORY
3681  016310  000417                        BR      9$              ;  ELSE TRY SOMEMORE
3682
3683  016312  022702  060000          7$:   CMP     #60000,R2       ;IF NOT POSSIBLY THE I/O PAGE
3684  016316  001014                        BNE     9$              ;  THEN BRANCH AND TRY SOMEMORE
3685  016320  005767  162212                TST     MMAVA           ;  ELSE IF 22 BIT ADDRESSING AVAILABLE
3686  016324  100405                        BMI     8$              ;     THEN BRANCH
3687  016326  022737  007600  172346        CMP     #7600,@#KIPAR3  ;     ELSE IF 128K I/O PAGE
3688  016334  001444                        BEQ     13$             ;        THEN ALL DONE
3689  016336  000404                        BR      9$              ;        ELSE TRY SOMEMORE
3690
3691  016340  022737  177600  172346  8$:   CMP     #177600,@#KIPAR3        ;IF 2M I/O PAGE
3692  016346  001437                        BEQ     13$             ;  THEN ALL DONE CHECKING
3693
3694  016350  032702  037777          9$:   BIT     #MASK8K,R2      ;IF NOT DONE WITH THIS 8K BANK
3695  016354  001303                        BNE     2$              ;  THEN TRY ANOTHER ADDRESS
3696
3697  016356  006300                  10$:  ASL     R0              ;IF NOT DONE WITH 128K BANK
3698  016360  001012                        BNE     11$             ;  THEN BRANCH
3699  016362  062701  000002                ADD     #2,R1           ;  ELSE POINT TO NEXT PMEMAP ENTRY
3700  016366  016700  163420                MOV     .PMEMAP,R0      ;START OF TABLE
3701  016372  062700  000040                ADD     #40,R0          ;END OF TABLE
3702  016376  020001                        CMP     R0,R1           ;IF DONE WITH TABLE
3703  016400  001422                        BEQ     13$             ;  THEN ALL DONE
```

```
3704  016402  012700  000001          MOV     #BIT0,R0        ;  ELSE REINIT BANK POINTER
3705
3706  016406  005767  162124  11$:    TST     MMAVA           ;IF KT AVAILABLE
3707  016412  001004                  BNE     12$             ;  THEN BRANCH
3708  016414  032700  000020          BIT     #BIT4,R0        ;  ELSE IF AT 32K LIMIT
3709  016420  001016                  BNE     14$             ;    THEN ALL DONE CHECKING
3710  016422  000653                  BR      1$              ;    ELSE TRY SOMEMORE
3711
3712  016424  062737  000400  172344  12$:    ADD     #400,@#KIPAR2   ;UPDATE PARS TO MAP TO NEXT
3713  016432  062737  000400  172346          ADD     #400,@#KIPAR3   ;  8K BANK
3714  016440  012702  040000          MOV     #40000,R2       ;REINIT ADDRESS
3715  016444  000642                  BR      1$              ;AND TRY SOMEMORE
3716
3717  016446                  13$:
3718  016446  012637  172346          MOV     (SP)+,@#KIPAR3          ;;POP STACK INTO @#KIPAR3
3719  016452  012637  172344          MOV     (SP)+,@#KIPAR2          ;;POP STACK INTO @#KIPAR2
3720
3721  016456  005704          14$:    TST     R4              ;IF ERROR FOUND
3722  016460  001003                  BNE     15$             ;  THEN BRANCH
3723  016462  004567  003106          JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
3724  016466  024026                  .WORD   NOPES           ;ADDRESS OF MESSAGE TO BE TYPED
3725
3726  016470                  15$:
3727  016470  012637  000116          MOV     (SP)+,@#116     ;;POP STACK INTO @#116
3728  016474  012637  000114          MOV     (SP)+,@#114     ;;POP STACK INTO @#114
3729  016500  012604                  MOV     (SP)+,R4        ;;POP STACK INTO R4
3730  016502  012603                  MOV     (SP)+,R3        ;;POP STACK INTO R3
3731  016504  012602                  MOV     (SP)+,R2        ;;POP STACK INTO R2
3732  016506  012601                  MOV     (SP)+,R1        ;;POP STACK INTO R1
3733  016510  012600                  MOV     (SP)+,R0        ;;POP STACK INTO R0
3734
3735  016512  000207          16$:    RTS     PC              ;RETURN
3736
3737
3738                          ;;*****************************************************************
3739                          ;ROUTINE TO CLEAR ALL PARITY REGISTERS PRESENT
3740                          ;;*****************************************************************
3741  016514                  CLRPAR:
3742  016514  010346                  MOV     R3,-(SP)        ;;PUSH R3 ON STACK
3743  016516  016703  163300          MOV     .MPRX,  R3      ;GET PARITY REGISTER TABLE POINTER.
3744  016522  005713          1$:     TST     (R3)            ;CHECK FOR THE TABLE TERMINATOR.
3745  016524  001402                  BEQ     2$              ;BR IF DONE ALL PARITY REGISTERS.
3746  016526  005033                  CLR     @(R3)+          ;CLEAR THE PARITY REGISTER.
3747  016530  000774                  BR      1$              ;BR FOR MORE
3748  016532                  2$:
3749  016532  012603                  MOV     (SP)+,R3        ;;POP STACK INTO R3
3750  016534  000207                  RTS     PC              ;RETURN.
3751
3752
3753                          .SBTTL  SUBROUTINES TO SET UP DATA FOR ERROR PRINTOUT ROUTINE.
3754                          ;;*****************************************************************
3755                          ;*  THESE ROUTINES ARE USED TO TRANSFER DATA TO COMMON TAG AREA (.$CMTAG)
3756                          ;*  FOR ERROR PRINTOUT BY .$ERROR & .$ERRTYP ROUTINES FROM **SYSMAC**.
3757                          ;;*****************************************************************
3758  016536  010267  162346  SPRNT:  MOV     R2,     $GDADR  ;SAVE THE ADDRESS UNDER TEST.
3759  016542  005067  162346          CLR     $GDDAT          ;SHOULD BE DATA IS '0'.
```

```
3760  016546  000435                        BR      SPRNTB
3761
3762  016550  014367  162374          SPRNTQ: MOV     -(R3),  $TMP0   ;GET THE PARITY REGISTER ADDRESS.
3763  016554  013367  162372                  MOV     @(R3)+, $TMP1   ;GET THE CONTENTS OF THE PARITY REG.
3764  016560  016767  163204  162330          MOV     TEMP,$BDDAT
3765  016566  010267  162316                  MOV     R2,$GDADR
3766  016572  000425                        BR      SPREND
3767
3768  016574  011367  162350          SPRNTP: MOV     (R3),   $TMP0   ;GET THE PARITY REGISTER ADDRESS.
3769  016600  010267  162304          SPRNT0: MOV     R2,     $GDADR  ;GET THE MEMORY ADDRESS BEING TESTED
3770  016604  000414                        BR      SPRNTA          ;BR TO COMMON SECTION.
3771
3772  016606  010267  162276          SPRNT1: MOV     R2,     $GDADR  ;GET THE MEMORY ADDRESS BEING TESTED
3773  016612  005367  162272                  DEC     $GDADR          ;ADJUST IT FOR PRINTOUT.
3774  016616  000407                        BR      SPRNTA          ;BR TO COMMON SECTION.
3775
3776  016620  010367  162324          SPRNT3: MOV     R3,     $TMP0   ;GET THE DATA IN R3.
3777  016624  010267  162260          SPRNT2: MOV     R2,     $GDADR  ;GET THE MEMORY ADDRESS BEING TESTED
3778  016630  162767  000002  162252          SUB     #2,     $GDADR  ;ADJUST IT FOR PRINTOUT.
3779  016636  010067  162252          SPRNTA: MOV     R0,     $GDDAT  ;GET WHAT THE DATA SHOULD BE
3780  016642  010167  162250          SPRNTB: MOV     R1,     $BDDAT  ;GET WHAT THE DATA WAS
3781  016646  000207                  SPREND: RTS     PC              ;RETURN TO ENTER ERROR ROUTINES
3782
3783
3784                                  ;;*****************************************************************
3785                                  ;*  THIS ROUTINE WILL TYPE OUT A MAP OF MEMORY BANKS.
3786                                  ;*  UPON ENTERING THIS ROUTINE R0 WILL CONTAIN THE ADDRESS
3787                                  ;*  OF THE TABLE TO BE TYPED OUT.  THE ROUTINE REQUIRES THAT THE
3788                                  ;*  TABLE CONTAIN 16 DECIMAL WORDS.
3789                                  ;*      STORAGE USED:
3790                                  ;*              R0      = MAP TABLE ADDRESS
3791                                  ;*              R1      = BANK POINTER
3792                                  ;*              R2      = CONSECUTIIVE ZERO BANK FLAG
3793                                  ;*              R3      - HIGH MEMORY ADR ... LO 16 BITS
3794                                  ;*              R4      = HIGH MEMORY ADR ... HI 6 BITS
3795                                  ;*              LMEMLO  = LOW MEMORY ADR ... LO 16 BITS
3796                                  ;*              LMEMHI  = LOW MEMORY ADR ... HI 6 BITS
3797                                  ;*              BNKTAB  = ADDRESS OF TABLE TO BE PROCESSED
3798                                  ;*              ENDBKT  = LAST ADDRESS + 2 OF TABLE TO BE PROCESSED
3799                                  ;*              FLG30K  = 30K MEMORY FLAG
3800                                  ;;*****************************************************************
3801  016650  010037  017350          TYPMAP: MOV     R0,@#BNKTAB     ;SAVE ADDRESS OF TABLE
3802  016654  012737  000040  017352          MOV     #40,@#ENDBKT    ;LOAD END BANK TABLE WITH 40
3803  016662  060037  017352                  ADD     R0,@#ENDBKT     ;GET LAST ADR + 2 INTO ENDBKT
3804  016666  005710                  1$:     TST     (R0)            ;ANY MEMORY IN THIS TABLE ENTRY
3805  016670  001011                          BNE     2$              ;  THEN GO PRINT OUT LOCATIONS
3806  016672  005720                          TST     (R0)+           ;  ELSE POINT TO NEXT TABLE ENTRY
3807  016674  023700  017352                  CMP     @#ENDBKT,R0     ;IF NOT END OF TABLE
3808  016700  001372                          BNE     1$              ;  THEN TRY AGAIN
3809  016702  004567  002666                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
3810  016706  024252                          .WORD   NOMEM           ;ADDRESS OF MESSAGE TO BE TYPED
3811                                                                  ;'NO MEMORY FOUND.'
3812  016710  000167  000426                  JMP     13$             ;EXIT
3813
3814  016714  013700  017350          2$:     MOV     @#BNKTAB,R0     ;RESTORE MAP TABLE ADDRESS
3815  016720  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
```

```
3816  016722  010246                      MOV     R2,-(SP)        ;;PUSH R2 ON STACK
3817  016724  010346                      MOV     R3,-(SP)        ;;PUSH R3 ON STACK
3818  016726  010446                      MOV     R4,-(SP)        ;;PUSH R4 ON STACK
3819  016730  005067  000410              CLR     LMEMLO          ;INIT LOW MEM ADR ... LO 16 BITS
3820  016734  005067  000406              CLR     LMEMHI          ;INIT LOW MEM ADR ... HI 6 BITS
3821  016740  005067  163024              CLR     TEMP            ;INIT LAST BANK IN MAP FLAG
3822  016744  012703  177777              MOV     #-1,R3          ;INIT HIGH MEM ADR ... LO 16 BITS
3823  016750  010304                      MOV     R3,R4           ;INIT HIGH MEM ADR ... HI 6 BITS
3824  016752  012702  000001              MOV     #1,R2           ;INIT CONSECUTIVE ZERO BANK FLAG
3825
3826  016756  012701  000001      3$:     MOV     #BIT0,R1        ;INIT BANK POINTER
3827
3828  016762  030110              4$:     BIT     R1,(R0)         ;IF THIS 8K BANK IS NOT PRESENT
3829  016764  001405                      BEQ     5$              ;  THEN GO PRINTOUT MAP
3830  016766  062703  040000              ADD     #40000,R3       ;  ELSE UPDATE HIGH MEM TO TOP OF THIS BANK
3831  016772  005504                      ADC     R4              ;
3832  016774  005002                      CLR     R2              ;CLEAR CONSECUTIVE ZERO BANK FLAG
3833  016776  000541                      BR      11$             ;GO CHECK FOR NEXT BANK
3834
3835  017000  005702              5$:     TST     R2              ;IF CONSECUTIVE ZERO BANK
3836  017002  001123                      BNE     10$             ;  THEN DO NOT PRINT MESSAGE
3837  017004  005202                      INC     R2              ;  ELSE UPDATE CONSECUTIVE ZERO BANK FLAG
3838  017006  022703  177777              CMP     #177777,R3      ;IF NOT 32K, 128K OR 2M BOUNDARY
3839  017012  001023                      BNE     9$              ;  THEN GO PRINT MESSAGE
3840  017014  005767  161516              TST     MMAVA           ;  ELSE IF 22BIT ADDRESSING
3841  017020  100413                      BMI     7$              ;    THEN GO WORK ON 22 BIT ADDRESSING
3842  017022  001006                      BNE     6$              ;    OR GO WORK ON 18 BIT ADDRESSING
3843  017024  005767  162744              TST     FLG30K          ;    ELSE IF NOT 30K SYSTEM
3844  017030  001412                      BEQ     8$              ;      THEN GO ADJ TO 28K BOUNDARY
3845  017032  162703  010000              SUB     #10000,R3       ;      ELSE ADJ TO 30K BOUNDARY
3846  017036  000411                      BR      9$              ;AND GO PRINT
3847
3848  017040  022704  000003      6$:     CMP     #3,R4           ;IF NOT 128K BOUNDARY
3849  017044  001006                      BNE     9$              ;  THEN GO PRINT
3850  017046  000403                      BR      8$              ;  ELSE GO ADJ TO 124K BOUNDARY
3851
3852  017050  022704  000077      7$:     CMP     #77,R4          ;IF NOT 2M BOUNDARY
3853  017054  001002                      BNE     9$              ;  THEN GO PRINT
3854
3855  017056  162703  020000      8$:     SUB     #20000,R3       ;ADJ HIGH MEMORY TO BELOW I/O PAGE
3856
3857  017062                      9$:
3858  017062  010346                      MOV     R3,-(SP)        ;;PUSH R3 ON STACK
3859  017064  010446                      MOV     R4,-(SP)        ;;PUSH R4 ON STACK
3860  017066  016746  000252              MOV     LMEMLO,-(SP)    ;;PUSH LMEMLO ON STACK
3861  017072  016746  000250              MOV     LMEMHI,-(SP)    ;;PUSH LMEMHI ON STACK
3862  017076  004567  002472              JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
3863  017102  023627                      .WORD   FROM            ;ADDRESS OF MESSAGE TO BE TYPED
3864  017104  006303                      ASL     R3              ;SHIFT BIT 15 INTO CARRY
3865  017106  006104                      ROL     R4              ;SHIFT BIT 15 TO ALIGN WITH BITS 17-16
3866  017110  006003                      ROR     R3              ;RESTORE BITS 14-0
3867  017112  006367  000226              ASL     LMEMLO          ;SHIFT BIT 15 INTO CARRY
3868  017116  006167  000224              ROL     LMEMHI          ;SHIFT BIT 15 TO ALIGN WITH BITS 17-16
3869  017122  006067  000216              ROR     LMEMLO          ;RESTORE BITS 14-0
3870  017126  016746  000214              MOV     LMEMHI,-(SP)    ;;SAVE LMEMHI FOR TYPEOUT
3871                                                              ;;TYPE ADDRESS BITS 21-15
```

```
3872                                    ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOS ROUTINE
3873                                    ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
3874  017132  106746                       MFPS    -(SP)           ;PUT THE PROCESSOR STATUS ON THE STACK
3875  017134  105066  000001              CLRB    1(SP)           ;HIGH BYTE CLEARED TO INSURE KERNEL MODE
3876                                                                ;ON PSW RETURN.
3877  017140  004767  003656              JSR     PC,     $TYPOS  ;GO TO THE SUBROUTINE
3878  017144     003                      .BYTE   3               ;;TYPE 3 DIGIT(S)
3879  017145     000                      .BYTE   0               ;;SUPPRESS LEADING ZEROS
3880  017146  016746  000172              MOV     LMEMLO,-(SP)    ;;SAVE LMEMLO FOR TYPEOUT
3881                                                                ;;TYPE ADDRESS BITS 14-0
3882                                    ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOS ROUTINE
3883                                    ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
3884  017152  106746                       MFPS    -(SP)           ;PUT THE PROCESSOR STATUS ON THE STACK
3885  017154  105066  000001              CLRB    1(SP)           ;HIGH BYTE CLEARED TO INSURE KERNEL MODE
3886                                                                ;ON PSW RETURN.
3887  017160  004767  003636              JSR     PC,     $TYPOS  ;GO TO THE SUBROUTINE
3888  017164     005                      .BYTE   5               ;;TYPE 5 DIGIT(S)
3889  017165     001                      .BYTE   1               ;;TYPE LEADING ZEROS
3890  017166  004567  002402              JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
3891  017172  023637                       .WORD   TO              ;ADDRESS OF MESSAGE TO BE TYPED
3892  017174  010446                       MOV     R4,-(SP)        ;;SAVE R4 FOR TYPEOUT
3893                                                                ;;TYPE ADDRESS BITS 21-15
3894                                    ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOS ROUTINE
3895                                    ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
3896  017176  106746                       MFPS    -(SP)           ;PUT THE PROCESSOR STATUS ON THE STACK
3897  017200  105066  000001              CLRB    1(SP)           ;HIGH BYTE CLEARED TO INSURE KERNEL MODE
3898                                                                ;ON PSW RETURN.
3899  017204  004767  003612              JSR     PC,     $TYPOS  ;GO TO THE SUBROUTINE
3900  017210     003                      .BYTE   3               ;;TYPE 3 DIGIT(S)
3901  017211     000                      .BYTE   0               ;;SUPPRESS LEADING ZEROS
3902  017212  010346                       MOV     R3,-(SP)        ;;SAVE R3 FOR TYPEOUT
3903                                                                ;;TYPE ADDRESS BITS 14-0
3904                                    ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOS ROUTINE
3905                                    ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
3906  017214  106746                       MFPS    -(SP)           ;PUT THE PROCESSOR STATUS ON THE STACK
3907  017216  105066  000001              CLRB    1(SP)           ;HIGH BYTE CLEARED TO INSURE KERNEL MODE
3908                                                                ;ON PSW RETURN.
3909  017222  004767  003574              JSR     PC,     $TYPOS  ;GO TO THE SUBROUTINE
3910  017226     005                      .BYTE   5               ;;TYPE 5 DIGIT(S)
3911  017227     001                      .BYTE   1               ;;TYPE LEADING ZEROS
3912  017230  012667  000112              MOV     (SP)+,LMEMHI    ;;POP STACK INTO LMEMHI
3913  017234  012667  000104              MOV     (SP)+,LMEMLO    ;;POP STACK INTO LMEMLO
3914  017240  012604                       MOV     (SP)+,R4        ;;POP STACK INTO R4
3915  017242  012603                       MOV     (SP)+,R3        ;;POP STACK INTO R3
3916  017244  005767  162520              TST     TEMP            ;HAS LAST BIT IN MAP PRINTED?
3917  017250  001030                       BNE     14$             ;YES,EXIT TO STACK CLEANUP
3918
3919  017252  062703  040000              10$:    ADD     #40000,R3       ;UPGRADE TO TOP OF THIS BANK
3920  017256  005504                       ADC     R4              ;
3921  017260  010367  000060              MOV     R3,LMEMLO       ;UPDATE LMEM TO THIS BANK
3922  017264  010467  000056              MOV     R4,LMEMHI       ;
3923  017270  062767  000001  000046      ADD     #1,LMEMLO       ;GET TO NEW LOW ADDRESS OF NEXT BANK
3924  017276  005567  000044              ADC     LMEMHI          ;
3925
3926  017302  006301                       11$:    ASL     R1              ;POINT TO NEXT 8K BANK
3927  017304  001226                       BNE     4$              ;BRANCH IF MORE TO TEST IN THIS MAP ENTRY
```

```
3928
3929  017306  005720                  12$:    TST     (R0)+           ;UPDATE POINTER TO NEXT TABLE ENTRY
3930  017310  023700  017352                  CMP     @#ENDBKT,R0     ;IF NOT DONE WITH TABLE
3931  017314  001220                          BNE     3$              ;  THEN TRY AGAIN
3932  017316  005760  177776                  TST     -2(R0)          ;CHECK IF LAST BANK AVAILABLE
3933  017322  100003                          BPL     14$             ;IF ZERO,TYPMAP PRINT COMPLETED
3934  017324  005267  162440                  INC     TEMP            ;ELSE SET LAST BANK PRINT FLAG
3935  017330  000623                          BR      5$              ;PRINT FOR LAST BANK
3936  017332                          14$:
3937  017332  012604                          MOV     (SP)+,R4        ;;POP STACK INTO R4
3938  017334  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
3939  017336  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
3940  017340  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
3941
3942  017342  000207                  13$:    RTS     PC              ;RETURN TO CALLER
3943
3944  017344  000000                  LMEMLO: .WORD   0               ; LOW MEMORY ADR ... LO 16 BITS
3945  017346  000000                  LMEMHI: .WORD   0               ; LOW MEMORY ADR ... HI 6 BITS
3946  017350  000000                  BNKTAB: .WORD   0               ; ADDRESS OF TABLE TO BE PROCESSED
3947  017352  000000                  ENDBKT: .WORD   0               ; LAST ADDRESS + 2 OF TABLE
3948                                                                  ; TO BE PROCESSED.
3949
3950                                  .SBTTL  SCOPE HANDLER ROUTINE
3951
3952                                  ;;**********************************************************************
3953                                  ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
3954                                  ;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
3955                                  ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
3956                                  ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
3957                                  ;*SW14=1         LOOP ON TEST
3958                                  ;*SW11=1         INHIBIT ITERATIONS
3959                                  ;*SW09=1         LOOP ON ERROR
3960                                  ;*SW08=1         LOOP ON TEST IN SWR<4:0>
3961                                  ;*CALL
3962                                  ;*      SCOPE           ;;SCOPE=IOT
3963
3964  017354                          $SCOPE:
3965                                  ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $CKSWR ROUTINE
3966                                  ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
3967  017354  106746                          MFPS    -(SP)           ;PUT THE PROCESSOR STATUS ON THE STACK
3968  017356  105066  000001                  CLRB    1(SP)           ;HIGH BYTE CLEARED TO INSURE KERNEL MODE
3969                                                                  ;ON PSW RETURN.
3970  017362  004767  001042                  JSR     PC,     $CKSWR  ;GO TO THE SUBROUTINE
3971  017366  010516                          MOV     R5,     (SP)    ;PUT RETURN PC ONTO STACK, SIMULATE JSR PC.
3972  017370  032777  040000  161532  1$:     BIT     #BIT14,@SWR     ;;LOOP ON PRESENT TEST?
3973  017376  001117                          BNE     $OVER           ;;YES IF SW14=1
3974                                  ;#####START OF CODE FOR THE XOR TESTER#####
3975  017400  000416                  $XTSTR: BR      6$              ;;IF RUNNING ON THE "XOR" TESTER CHANGE
3976                                                                  ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
3977  017402  013746  000004                  MOV     @#ERRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
3978  017406  012737  017426  000004          MOV     #5$,@#ERRVEC    ;;SET FOR TIMEOUT
3979  017414  005737  177060                  TST     @#177060        ;;TIME OUT ON XOR?
3980  017420  012637  000004                  MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
3981  017424  000466                          BR      $SVLAD          ;;GO TO THE NEXT TEST
3982  017426  022626                  5$:     CMP     (SP)+,(SP)+     ;;CLEAR THE STACK AFTER A TIME OUT
3983  017430  012637  000004                  MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
```

```
 3984  017434  000426                          BR      7$              ;;LOOP ON THE PRESENT TEST
 3985  017436                          6$:;#####END OF CODE FOR THE XOR TESTER#####
 3986  017436  032777  000400  161464          BIT     #BIT08,@SWR     ;;LOOP ON SPEC. TEST?
 3987  017444  001407                          BEQ     2$              ;;BR IF NO
 3988  017446  017746  161456                  MOV     @SWR,-(SP)      ;;SET DESIRED TEST NUM. FROM SWR
 3989  017452  042716  000340                  BIC     #$SWRMK,(SP)    ;;STRIP AWAY UNDESIRED BITS
 3990  017456  122667  161410                  CMPB    (SP)+,$TSTNM    ;;ON THE RIGHT TEST?
 3991  017462  001465                          BEQ     $OVER           ;;BR IF YES
 3992  017464  105767  161403          2$:     TSTB    $ERFLG          ;;HAS AN ERROR OCCURRED?
 3993  017470  001421                          BEQ     3$              ;;BR IF NO
 3994  017472  126767  161407  161373          CMPB    $ERMAX,$ERFLG   ;;MAX. ERRORS FOR THIS TEST OCCURRED?
 3995  017500  101015                          BHI     3$              ;;BR IF NO
 3996  017502  032777  001000  161420          BIT     #BIT09,@SWR     ;;LOOP ON ERROR?
 3997  017510  001404                          BEQ     4$              ;;BR IF NO
 3998  017512  016767  161362  161356  7$:     MOV     $LPERR,$LPADR   ;;SET LOOP ADDRESS TO LAST SCOPE
 3999  017520  000446                          BR      $OVER
 4000  017522  105067  161345          4$:     CLRB    $ERFLG          ;;ZERO THE ERROR FLAG
 4001  017526  005067  161426                  CLR     $TIMES          ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
 4002  017532  000415                          BR      1$              ;;ESCAPE TO THE NEXT TEST
 4003  017534  032777  004000  161366  3$:     BIT     #BIT11,@SWR     ;;INHIBIT ITERATIONS?
 4004  017542  001011                          BNE     1$              ;;BR IF YES
 4005  017544  005767  161432                  TST     $PASS           ;;IF FIRST PASS OF PROGRAM
 4006  017550  001406                          BEQ     1$              ;;      INHIBIT ITERATIONS
 4007  017552  005267  161316                  INC     $ICNT           ;;INCREMENT ITERATION COUNT
 4008  017556  026767  161376  161310          CMP     $TIMES,$ICNT    ;;CHECK THE NUMBER OF ITERATIONS MADE
 4009  017564  002024                          BGE     $OVER           ;;BR IF MORE ITERATION REQUIRED
 4010  017566  012767  000001  161300  1$:     MOV     #1,$ICNT        ;;REINITIALIZE THE ITERATION COUNTER
 4011  017574  016767  000052  161356          MOV     $MXCNT,$TIMES   ;;SET NUMBER OF ITERATIONS TO DO
 4012  017602  105267  161264          $SVLAD: INCB    $TSTNM          ;;COUNT TEST NUMBERS
 4013  017606  116767  161260  161364          MOVB    $TSTNM,$TESTN   ;;SET TEST NUMBER IN APT MAILBOX
 4014  017614  011667  161256                  MOV     (SP),$LPADR     ;;SAVE SCOPE LOOP ADDRESS
 4015  017620  011667  161254                  MOV     (SP),$LPERR     ;;SAVE ERROR LOOP ADDRESS
 4016  017624  005067  161332                  CLR     $ESCAPE         ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
 4017  017630  112767  000001  161247          MOVB    #1,$ERMAX       ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
 4018  017636  016777  161230  161266  $OVER:  MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
 4019  017644  016716  161226                  MOV     $LPADR,(SP)     ;;FUDGE RETURN ADDRESS
 4020  017650  000207                  ENDINS: RTS     PC              ;EXIT SCOPE ROUTINE BACK TO TEST.
 4021  017652  000001                  $MXCNT: 1                       ;;MAX. NUMBER OF ITERATIONS
 4022                                  ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $CKSWR ROUTINE
 4023                                  ;*  WIHTOUT USING A ''TRAP'' INSTRUCTION AS CALLED FOR BY **SYSMAC**.
 4024  017654  106746                          MFPS    -(SP)           ;PUT THE PROCESSOR STATUS ON THE STACK
 4025  017656  105066  000001                  CLRB    1(SP)           ;HIGH BYTE CLEARED TO INSURE KERNEL MODE
 4026                                                                  ;ON PSW RETURN.
 4027  017662  004767  000542                  JSR     PC,     $CKSWR  ;GO TO THE SUBROUTINE
 4028                                  .SBTTL  ERROR HANDLER ROUTINE
 4029
 4030                                  ;;**********************************************************************
 4031                                  ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
 4032                                  ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
 4033                                  ;*AND GO TO $ERRTYP ON ERROR
 4034                                  ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
 4035                                  ;*SW15=1         HALT ON ERROR
 4036                                  ;*SW13=1         INHIBIT ERROR TYPEOUTS
 4037                                  ;*SW10-1         BELL ON ERROR
 4038                                  ;*SW09=1         LOOP ON ERROR
 4039                                  ;*CALL
```

```
                                        ;*      ERROR   N       ;;ERROR=EMT AND N=ERROR ITEM NUMBER
                                        $ERROR:
                                        ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $CKSWR ROUTINE
                                        ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
017666  106746                          MFPS    -(SP)           ;PUT THE PROCESSOR STATUS ON THE STACK
017670  105066  000001                  CLRB    1(SP)           ;HIGH BYTE CLEARED TO INSURE KERNEL MODE
                                                                ;ON PSW RETURN.
017674  004767  000530                  JSR     PC,     $CKSWR  ;GO TO THE SUBROUTINE
017700  062716  000002                  ADD     #2,     (SP)    ;ADJUST POINTER PAST CODE WORD.
017704  105267  161163          7$:     INCB    $ERFLG          ;;SET THE ERROR FLAG
017710  001775                          BEQ     7$              ;;DON'T LET THE FLAG GO TO ZERO
017712  016777  161154  161212          MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
017720  032777  002000  161202          BIT     #BIT10,@SWR     ;;BELL ON ERROR?
017726  001403                          BEQ     1$              ;;NO - SKIP
017730  004567  001640                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
017734  001164                          .WORD   $BELL           ;ADDRESS OF MESSAGE TO BE TYPED
017736  005267  161140          1$:     INC     $ERTTL          ;;COUNT THE NUMBER OF ERRORS
017742  011667  161140                  MOV     (SP),$ERRPC     ;;GET ADDRESS OF ERROR INSTRUCTION
017746  162767  000002  161132          SUB     #2,$ERRPC
017754  117767  161126  161122          MOVB    @$ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
017762  032777  020000  161140          BIT     #BIT13,@SWR     ;;SKIP TYPEOUT IF SET
017770  001005                          BNE     20$             ;;SKIP TYPEOUTS
017772  004767  000120                  JSR     PC,$ERRTYP      ;;GO TO USER ERROR ROUTINE
017776  004567  001572                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
020002  001171                          .WORD   $CRLF           ;ADDRESS OF MESSAGE TO BE TYPED
020004                          20$:
020004  122767  000001  161202          CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
020012  001007                          BNE     2$              ;;NO,SKIP APT ERROR REPORT
020014  116767  161064  000004          MOVB    $ITEMB,21$      ;;SET ITEM NUMBER AS ERROR NUMBER
020022  004767  002152                  JSR     PC,$ATY4        ;;REPORT FATAL ERROR TO APT
020026     000          21$:    .BYTE   0
020027     000                  .BYTE   0
020030  000777          22$:    BR      22$             ;;APT ERROR LOOP
020032  005777  161072  2$:     TST     @SWR            ;;HALT ON ERROR
020036  100006                  BPL     3$              ;;SKIP IF CONTINUE
020040  000000                  HALT                    ;;HALT ON ERROR!
                                        ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $CKSWR ROUTINE
                                        ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
020042  106746                          MFPS    -(SP)           ;PUT THE PROCESSOR STATUS ON THE STACK
020044  105066  000001                  CLRB    1(SP)           ;HIGH BYTE CLEARED TO INSURE KERNEL MODE
                                                                ;ON PSW RETURN.
020050  004767  000354                  JSR     PC,     $CKSWR  ;GO TO THE SUBROUTINE
020054  032777  001000  161046  3$:     BIT     #BIT09,@SWR     ;;LOOP ON ERROR SWITCH SET?
020062  001402                          BEQ     4$              ;;BR IF NO
020064  016716  161010                  MOV     $LPERR,(SP)     ;;FUDGE RETURN FOR LOOPING
020070  005767  161066          4$:     TST     $ESCAPE         ;;CHECK FOR AN ESCAPE ADDRESS
020074  001402                          BEQ     5$              ;;BR IF NONE
020076  016716  161060                  MOV     $ESCAPE,(SP)    ;;FUDGE RETURN ADDRESS FOR ESCAPE
020102                          5$:
020102  022737  012560  000042          CMP     #$ENDAD,@#42    ;;ACT-11 AUTO-ACCEPT?
020110  001001                          BNE     6$              ;;BRANCH IF NO
020112  000000                          HALT                    ;;YES
020114                          6$:
020114  000207                          RTS     PC
                                        ;;*****************************************************************
```

```
4096
4097                                    .SBTTL  ERROR MESSAGE TYPEOUT ROUTINE
4098
4099
4100
4101                                    ;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
4102                                    ;*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
4103                                    ;*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
4104
4105  020116                            $ERRTYP:
4106  020116  004567  001452                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
4107  020122  001171                          .WORD   $CRLF           ;ADDRESS OF MESSAGE TO BE TYPED
4108  020124  010046                          MOV     R0,-(SP)                ;SAVE R0
4109  020126  005000                          CLR     R0                      ;PICKUP THE ITEM INDEX
4110  020130  156700  160750                BISB    $ITEMB,R0
4111  020134  001010                          BNE     1$                      ;IF ITEM NUMBER IS ZERO, JUST
4112                                                                          ;TYPE THE PC OF THE ERROR
4113  020136  016746  160744                MOV     $ERRPC,-(SP)    ;;SAVE $ERRPC FOR TYPEOUT
4114                                                                  ;;ERROR ADDRESS
4115                                    ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOC ROUTINE
4116                                    ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
4117  020142  106746                          MFPS    -(SP)           ;PUT THE PROCESSOR STATUS ON THE STACK
4118  020144  105066  000001                CLRB    1(SP)           ;HIGH BYTE CLEARED TO INSURE KERNEL MODE
4119                                                                  ;ON PSW RETURN.
4120  020150  004767  002672                JSR     PC,     $TYPOC  ;GO TO THE SUBROUTINE
4121  020154  000516                          BR      10$                     ;GET OUT
4122  020156  016767  160724  161320  1$:   MOV     $ERRPC, $VERPC  ;SET UP VIRTUAL PC FOR TYPEOUT.
4123  020164  166767  160342  161312        SUB     RELOCF, $VERPC  ;MAKE VIRTUAL IF NOT ALREADY.
4124  020172  005300                          DEC     R0                      ;ADJUST THE INDEX SO THAT IT WILL
4125  020174  006300                          ASL     R0                      ;       WORK FOR THE ERROR TABLE
4126  020176  006300                          ASL     R0
4127  020200  006300                          ASL     R0
4128  020202  066700  161624                ADD     .ERRTB, R0      ;FORM TABLE POINTER
4129  020206  012067  000006                MOV     (R0)+,2$                ;PICKUP "ERROR MESSAGE" POINTER
4130  020212  001406                          BEQ     3$                      ;SKIP TYPEOUT IF NO POINTER
4131  020214  004567  001354                JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
4132  020220  000000                2$:       .WORD   0                       ;"ERROR MESSAGE" POINTER GOES HERE
4133  020222  004567  001346                JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
4134  020226  001171                          .WORD   $CRLF           ;ADDRESS OF MESSAGE TO BE TYPED
4135  020230  012067  000006        3$:       MOV     (R0)+,4$                ;PICKUP "DATA HEADER" POINTER
4136  020234  001406                          BEQ     5$                      ;SKIP TYPEOUT IF 0
4137  020236  004567  001332                JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
4138  020242  000000                4$:       .WORD   0                       ;"DATA HEADER" POINTER GOES HERE
4139  020244  004567  001324                JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
4140  020250  001171                          .WORD   $CRLF           ;ADDRESS OF MESSAGE TO BE TYPED
4141  020252  010146                5$:       MOV     R1,-(SP)                ;SAVE R1
4142  020254  012001                          MOV     (R0)+,R1                ;PICKUP "DATA TABLE" POINTER
4143  020256  001454                          BEQ     9$                      ;BR IF NO DATA TO BE TYPED
4144  020260  066701  160246                ADD     RELOCF, R1      ;ADJUST POINTER
4145  020264  012000                          MOV     (R0)+,R0                ;PICKUP "DATA FORMAT" POINTER
4146  020266  066700  160240                ADD     RELOCF, R0      ;ADJUST POINTER.
4147  020272  105720                6$:       TSTB    (R0)+                   ;CHECK THE FORMAT
4148  020274  001007                          BNE     7$                      ;BR IF NOT 16-BIT OCTAL
4149  020276  013146                          MOV     @(R1)+,-(SP)    ;;SAVE @(R1)+ FOR TYPEOUT
4150                                    ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOC ROUTINE
4151                                    ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
```

```
 4152  020300  106746                          MFPS    -(SP)           ;PUT THE PROCESSOR STATUS ON THE STACK
 4153  020302  105066  000001                  CLRB    1(SP)           ;HIGH BYTE CLEARED TO INSURE KERNEL MODE
 4154                                                                  ;ON PSW RETURN.
 4155  020306  004767  002534                  JSR     PC,     $TYPOC  ;GO TO THE SUBROUTINE
 4156  020312  000430                          BR      8$
 4157  020314  100407                  7$:     BMI     17$             ;BRANCH IF NOT DECIMAL
 4158  020316  013146                          MOV     @(R1)+,-(SP)    ;;SAVE @(R1)+ FOR TYPEOUT
 4159                                  ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPDS ROUTINE
 4160                                  ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
 4161  020320  106746                          MFPS    -(SP)           ;PUT THE PROCESSOR STATUS ON THE STACK
 4162  020322  105066  000001                  CLRB    1(SP)           ;HIGH BYTE CLEARED TO INSURE KERNEL MODE
 4163                                                                  ;ON PSW RETURN.
 4164  020326  004767  002236                  JSR     PC,     $TYPDS  ;GO TO THE SUBROUTINE
 4165  020332  000420                          BR      8$              ;SKIP
 4166  020334  122760  177777  177777  17$:     CMPB    #-1,    -1(R0)  ;CHECK FOR 22-BIT ADDRESS FORMAT.
 4167  020342  001004                          BNE     18$             ;BR IF NOT 22-BIT ADDRESS FORMAT.
 4168  020344  013146                          MOV     @(R1)+, -(SP)   ;PUT THE DATA ON THE STACK.
 4169  020346  004767  002736                  JSR     PC,     $TYPAD  ;DETERMINE THE PHYSICAL ADDRESS AND TYPE IT.
 4170  020352  000410                          BR      8$              ;SKIP
 4171  020354                          18$:
 4172  020354  005046                          CLR     -(SP)           ;CLEAR THE WORD ON THE STACK.
 4173  020356  113116                          MOVB    @(R1)+, (SP)    ;PUT THE DATA ON THE STACK.
 4174                                  ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOS ROUTINE
 4175                                  ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
 4176  020360  106746                          MFPS    -(SP)           ;PUT THE PROCESSOR STATUS ON THE STACK
 4177  020362  105066  000001                  CLRB    1(SP)           ;HIGH BYTE CLEARED TO INSURE KERNEL MODE
 4178                                                                  ;ON PSW RETURN.
 4179  020366  004767  002430                  JSR     PC,     $TYPOS  ;GO TO THE SUBROUTINE
 4180  020372     003                          .BYTE   3               ;TYPE 3 DIGITS.
 4181  020373     001                          .BYTE   1               ;TYPE LEADING ZEROS.
 4182  020374  005711                  8$:     TST     (R1)                    ;IS THERE ANOTHER NUMBER?
 4183  020376  001404                          BEQ     9$                      ;BR IF NO
 4184  020400  004567  001170                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 4185  020404  020424                          .WORD   11$             ;ADDRESS OF MESSAGE TO BE TYPED
 4186  020406  000731                          BR      6$                      ;LOOP
 4187
 4188  020410  012601                  9$:     MOV     (SP)+,R1                ;RESTORE R1
 4189  020412  012600                  10$:    MOV     (SP)+,R0                ;RESTORE R0
 4190  020414  004567  001154                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 4191  020420  001171                          .WORD   $CRLF           ;ADDRESS OF MESSAGE TO BE TYPED
 4192  020422  000207                          RTS     PC                      ;RETURN
 4193  020424  020040     000          11$:    .ASCIZ  /  /            ;DOUBLE SPACE
 4194          020430                          .EVEN
 4195                                  .SBTTL  TTY INPUT ROUTINE
 4196
 4197                                  ;;******************************************************************
 4198                                  .ENABL  LSB
 4199
 4200                                  ;;******************************************************************
 4201                                  ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
 4202                                  ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
 4203                                  ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
 4204                                  ;*WHEN OPERATING IN TTY FLAG MODE.
 4205  020430  022767  000176  160472  $CKSWR: CMP     #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED?
 4206  020436  001105                          BNE     15$             ;;BRANCH IF NO
 4207  020440  105777  160470                  TSTB    @$TKS           ;;CHAR THERE?
```

```
 4208  020444  100102                          BPL     15$             ;;IF NO, DON'T WAIT AROUND
 4209  020446  117746  160464                  MOVB    @$TKB,-(SP)     ;;SAVE THE CHAR
 4210  020452  042716  177600                  BIC     #^C177,(SP)     ;;STRIP-OFF THE ASCII
 4211  020456  022726  000007                  CMP     #7,(SP)+        ;;IS IT A CONTROL G?
 4212  020462  001073                          BNE     15$             ;;NO, RETURN TO USER
 4213  020464  126727  160434  000001          CMPB    $AUTOB,#1       ;;ARE WE RUNNING IN AUTO-MODE?
 4214  020472  001467                          BEQ     15$             ;;BRANCH IF YES
 4215
 4216  020474  004567  001074                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 4217  020500  021371                          .WORD   $CNTLG          ;ADDRESS OF MESSAGE TO BE TYPED
 4218  020502                          $GTSWR:
 4219  020502  004567  001066                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 4220  020506  021376                          .WORD   $MSWR           ;ADDRESS OF MESSAGE TO BE TYPED
 4221  020510  016746  157462                  MOV     SWREG,-(SP)     ;;SAVE SWREG FOR TYPEOUT
 4222                                  ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOC ROUTINE
 4223                                  ;*  WIHTOUT USING A ''TRAP'' INSTRUCTION AS CALLED FOR BY **SYSMAC**.
 4224  020514  106746                          MFPS    -(SP)           ;PUT THE PROCESSOR STATUS ON THE STACK
 4225  020516  105066  000001                  CLRB    1(SP)           ;HIGH BYTE CLEARED TO INSURE KERNEL MODE
 4226                                                                  ;ON PSW RETURN.
 4227  020522  004767  002320                  JSR     PC,     $TYPOC  ;GO TO THE SUBROUTINE
 4228  020526  004567  001042                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 4229  020532  021407                          .WORD   $MNEW           ;ADDRESS OF MESSAGE TO BE TYPED
 4230  020534  005046                  19$:    CLR     -(SP)           ;;CLEAR COUNTER
 4231  020536  005046                          CLR     -(SP)           ;;THE NEW SWR
 4232  020540  105777  160370          7$:     TSTB    @$TKS           ;;CHAR THERE?
 4233  020544  100375                          BPL     7$              ;;IF NOT TRY AGAIN
 4234
 4235  020546  117746  160364                  MOVB    @$TKB,-(SP)     ;;PICK UP CHAR
 4236  020552  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
 4237
 4238
 4239
 4240  020556  021627  000025          9$:     CMP     (SP),#25        ;;IS IT A CONTROL-U?
 4241  020562  001006                          BNE     10$             ;;BRANCH IF NOT
 4242  020564  004567  001004                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 4243  020570  021364                          .WORD   $CNTLU          ;ADDRESS OF MESSAGE TO BE TYPED
 4244  020572  062706  000006          20$:    ADD     #6,SP           ;;IGNORE PREVIOUS INPUT
 4245  020576  000756                          BR      19$             ;;LET'S TRY IT AGAIN
 4246
 4247
 4248  020600  021627  000015          10$:    CMP     (SP),#15        ;;IS IT A <CR>?
 4249  020604  001023                          BNE     16$             ;;BRANCH IF NO
 4250  020606  005766  000004                  TST     4(SP)           ;;YES, IS IT THE FIRST CHAR?
 4251  020612  001403                          BEQ     11$             ;;BRANCH IF YES
 4252  020614  016677  000002  160306          MOV     2(SP),@SWR      ;;SAVE NEW SWR
 4253  020622  062706  000006          11$:    ADD     #6,SP           ;;CLEAR UP STACK
 4254  020626                          14$:
 4255  020626  004567  000742                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 4256  020632  001171                          .WORD   $CRLF           ;ADDRESS OF MESSAGE TO BE TYPED
 4257  020634  126727  160265  000001          CMPB    $INTAG,#1       ;;RE-ENABLE TTY KBD INTERRUPTS?
 4258  020642  001003                          BNE     15$             ;;BRANCH IF NOT
 4259  020644  012777  000100  160262          MOV     #100,@$TKS      ;;RE-ENABLE TTY KBD INTERRUPTS
 4260  020652  000002                  15$:    RTI                     ;;RETURN
 4261  020654  004767  001160          16$:    JSR     PC,$TYPEC       ;;ECHO CHAR
 4262  020660  021627  000060                  CMP     (SP),#60        ;;CHAR < 0?
 4263  020664  002420                          BLT     18$             ;;BRANCH IF YES
```

```
4264  020666  021627  000067                CMP     (SP),#67        ;;CHAR > 7?
4265  020672  003015                        BGT     18$             ;;BRANCH IF YES
4266  020674  042726  000060                BIC     #60,(SP)+       ;;STRIP-OFF ASCII
4267  020700  005766  000002                TST     2(SP)           ;;IS THIS THE FIRST CHAR
4268  020704  001403                        BEQ     17$             ;;BRANCH IF YES
4269  020706  006316                        ASL     (SP)            ;;NO, SHIFT PRESENT
4270  020710  006316                        ASL     (SP)            ;;   CHAR OVER TO MAKE
4271  020712  006316                        ASL     (SP)            ;;   ROOM FOR NEW ONE.
4272  020714  005266  000002        17$:    INC     2(SP)           ;;KEEP COUNT OF CHAR
4273  020720  056616  177776                BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
4274  020724  000705                        BR      7$              ;;GET THE NEXT ONE
4275  020726                        18$:
4276  020726  004567  000642                JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
4277  020732  001170                        .WORD   $QUES           ;ADDRESS OF MESSAGE TO BE TYPED
4278  020734  000716                        BR      20$             ;;SIMULATE CONTROL-U
4279                                .DSABL  LSB
4280
4281
4282                                ;;*****************************************************************
4283                                ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
4284                                ;*CALL:
4285                                ;*      RDCHR                   ;;INPUT A SINGLE CHARACTER FROM THE TTY
4286                                ;*      RETURN HERE             ;;CHARACTER IS ON THE STACK
4287                                ;*                              ;;WITH PARITY BIT STRIPPED OFF
4288                                ;
4289
4290  020736  011646                $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC
4291  020740  016666  000004  000002        MOV     4(SP),2(SP)     ;;SAVE THE PS
4292  020746  105777  160162        1$:     TSTB    @$TKS           ;;WAIT FOR
4293  020752  100375                        BPL     1$              ;;A CHARACTER
4294  020754  117766  160156  000004        MOVB    @$TKB,4(SP)     ;;READ THE TTY
4295  020762  042766  177600  000004        BIC     #^C<177>,4(SP)  ;;GET RID OF JUNK IF ANY
4296  020770  026627  000004  000023        CMP     4(SP),#23       ;;IS IT A CONTROL-S?
4297  020776  001013                        BNE     3$              ;;BRANCH IF NO
4298  021000  105777  160130        2$:     TSTB    @$TKS           ;;WAIT FOR A CHARACTER
4299  021004  100375                        BPL     2$              ;;LOOP UNTIL ITS THERE
4300  021006  117746  160124                MOVB    @$TKB,-(SP)     ;;GET CHARACTER
4301  021012  042716  177600                BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
4302  021016  022627  000021                CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
4303  021022  001366                        BNE     2$              ;;IF NOT DISCARD IT
4304  021024  000750                        BR      1$              ;;YES, RESUME
4305  021026  026627  000004  000021  3$:   CMP     4(SP),#$XON     ;;IS IT A RANDOM XON?                   ;RAN001
4306  021034  001744                        BEQ     1$              ;;BRANCH IF YES                         ;RAN001
4307  021036  026627  000004  000140        CMP     4(SP),#140      ;;IS IT UPPER CASE?
4308  021044  002407                        BLT     4$              ;;BRANCH IF YES
4309  021046  026627  000004  000175        CMP     4(SP),#175      ;;IS IT A SPECIAL CHAR?
4310  021054  003003                        BGT     4$              ;;BRANCH IF YES
4311  021056  042766  000040  000004        BIC     #40,4(SP)       ;;MAKE IT UPPER CASE
4312  021064  000002                4$:     RTI                     ;;GO BACK TO USER
4313                                ;;*****************************************************************
4314                                ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
4315                                ;*CALL:
4316                                ;*      RDLIN                   ;;INPUT A STRING FROM THE TTY
4317                                ;*      RETURN HERE             ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
4318                                ;*                              ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
4319
```

```
4320  021066  010346                  $RDLIN: MOV     R3,-(SP)        ;;SAVE R3
4321  021070  005046                          CLR     -(SP)           ;;CLEAR THE RUBOUT KEY
4322  021072  012703  021354            1$:   MOV     #$TTYIN,R3      ;;GET ADDRESS
4323  021076  022703  021364            2$:   CMP     #$TTYIN+8.,R3   ;;BUFFER FULL?
4324  021102  101470                          BLOS    4$              ;;BR IF YES
4325                                    ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $RDCHR ROUTINE
4326                                    ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
4327  021104  106746                          MFPS    -(SP)           ;PUT THE PROCESSOR STATUS ON THE STACK
4328  021106  105066  000001                CLRB    1(SP)           ;HIGH BYTE CLEARED TO INSURE KERNEL MODE
4329                                                                  ;ON PSW RETURN.
4330  021112  004767  177620                JSR     PC,     $RDCHR  ;GO TO THE SUBROUTINE
4331  021116  112613                          MOVB    (SP)+,(R3)      ;;GET CHARACTER
4332  021120  122713  000177            10$:  CMPB    #177,(R3)       ;;IS IT A RUBOUT
4333  021124  001024                          BNE     5$              ;;BR IF NO
4334  021126  005716                          TST     (SP)            ;;IS THIS THE FIRST RUBOUT?
4335  021130  001010                          BNE     6$              ;;BR IF NO
4336  021132  112767  000134  000212        MOVB    #'\,9$          ;;TYPE A BACK SLASH
4337  021140  004567  000430                JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
4338  021144  021352                          .WORD   9$              ;ADDRESS OF MESSAGE TO BE TYPED
4339  021146  012716  177777                MOV     #-1,(SP)        ;;SET THE RUBOUT KEY
4340  021152  005303                    6$:   DEC     R3              ;;BACKUP BY ONE
4341  021154  020327  021354                CMP     R3,#$TTYIN      ;;STACK EMPTY?
4342  021160  103441                          BLO     4$              ;;BR IF YES
4343  021162  111367  000164                MOVB    (R3),9$         ;;SETUP TO TYPEOUT THE DELETED CHAR.
4344  021166  004567  000402                JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
4345  021172  021352                          .WORD   9$              ;ADDRESS OF MESSAGE TO BE TYPED
4346  021174  000740                          BR      2$              ;;GO READ ANOTHER CHAR.
4347  021176  005716                    5$:   TST     (SP)            ;;RUBOUT KEY SET?
4348  021200  001407                          BEQ     7$              ;;BR IF NO
4349  021202  112767  000134  000142        MOVB    #'\,9$          ;;TYPE A BACK SLASH
4350  021210  004567  000360                JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
4351  021214  021352                          .WORD   9$              ;ADDRESS OF MESSAGE TO BE TYPED
4352  021216  005016                          CLR     (SP)            ;;CLEAR THE RUBOUT KEY
4353  021220  122713  000025            7$:   CMPB    #25,(R3)        ;;IS CHARACTER A CTRL U?
4354  021224  001004                          BNE     8$              ;;BR IF NO
4355  021226  004567  000342                JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
4356  021232  021364                          .WORD   $CNTLU          ;ADDRESS OF MESSAGE TO BE TYPED
4357  021234  000716                          BR      1$              ;;GO START OVER
4358  021236  122713  000022            8$:   CMPB    #22,(R3)        ;;IS CHARACTER A '^R'?
4359  021242  001014                          BNE     3$              ;;BRANCH IF NO
4360  021244  105013                          CLRB    (R3)            ;;CLEAR THE CHARACTER
4361  021246  004567  000322                JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
4362  021252  001171                          .WORD   $CRLF           ;ADDRESS OF MESSAGE TO BE TYPED
4363  021254  004567  000314                JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
4364  021260  021354                          .WORD   $TTYIN          ;ADDRESS OF MESSAGE TO BE TYPED
4365  021262  000705                          BR      2$              ;;GO PICKUP ANOTHER CHACTER
4366  021264                            4$:
4367  021264  004567  000304                JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
4368  021270  001170                          .WORD   $QUES           ;ADDRESS OF MESSAGE TO BE TYPED
4369  021272  000677                          BR      1$              ;;CLEAR THE BUFFER AND LOOP
4370  021274  111367  000052            3$:   MOVB    (R3),9$         ;;ECHO THE CHARACTER
4371  021300  004567  000270                JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
4372  021304  021352                          .WORD   9$              ;ADDRESS OF MESSAGE TO BE TYPED
4373  021306  122723  000015                CMPB    #15,(R3)+       ;;CHECK FOR RETURN
4374  021312  001271                          BNE     2$              ;;LOOP IF NOT RETURN
4375  021314  105063  177777                CLRB    -1(R3)          ;;CLEAR RETURN (THE 15)
```

```
4376  021320  004567  000250                    JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
4377  021324  001172                            .WORD   $LF             ;ADDRESS OF MESSAGE TO BE TYPED
4378  021326  005726                            TST     (SP)+           ;;CLEAN RUBOUT KEY FROM THE STACK
4379  021330  012603                            MOV     (SP)+,R3        ;;RESTORE R3
4380  021332  011646                            MOV     (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
4381  021334  016666  000004  000002            MOV     4(SP),2(SP)     ;;      FIRST ASCII CHARACTER ON IT
4382  021342  012766  021354  000004            MOV     #$TTYIN,4(SP)
4383  021350  000002                            RTI                     ;;RETURN
4384  021352     000                  9$:       .BYTE   0               ;;STORAGE FOR ASCII CHAR. TO TYPE
4385  021353     000                            .BYTE   0               ;;TERMINATOR
4386  021354  000010               $TTYIN:     .BLKB   8.              ;;RESERVE 8 BYTES FOR TTY INPUT
4387  021364  052536  005015     000  $CNTLU:  .ASCIZ  /^U/<15><12>    ;;CONTROL 'U'
4388  021371     136  006507  000012  $CNTLG:  .ASCIZ  /^G/<15><12>    ;;CONTROL 'G'
4389  021376  005015  053523  020122  $MSWR:   .ASCIZ  <15><12>/SWR = /
4390  021404  020075     000
4391  021407     040  047040  053505  $MNEW:   .ASCIZ  /  NEW = /
4392  021414  036440  000040
4393                                    .SBTTL  READ AN OCTAL NUMBER FROM THE TTY
4394
4395                                    ;;******************************************************************
4396                                    ;*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
4397                                    ;*CHANGE IT TO BINARY.
4398                                    ;*THE INPUT CHARACTERS WILL BE CHECKED TO INSURED THEY ARE LEGAL
4399                                    ;*OCTAL DIGITS. IF AN ILLEGAL CHARACTER IS READ A "?" WILL BE TYPED
4400                                    ;*FOLLOWED BY A CARRIAGE RETURN-LINE FEED. THE COMPLETE NUMBER MUST
4401                                    ;*THEN BE RETYPED. THE INPUT IS TERMINATED BY TYPING A CARRIAGE RETURN.
4402                                    ;*CALL:
4403                                    ;*      RDOCT                   ;;READ AN OCTAL NUMBER
4404                                    ;*      RETURN HERE             ;;LOW ORDER BITS ARE ON TOP OF THE STACK
4405                                    ;*                              ;;HIGH ORDER BITS ARE IN $HIOCT
4406
4407  021420  011646               $RDOCT: MOV     (SP),-(SP)      ;;PROVIDE SPACE FOR THE
4408  021422  016666  000004  000002            MOV     4(SP),2(SP)     ;;INPUT NUMBER
4409  021430  010046                            MOV     R0,-(SP)        ;;PUSH R0 ON STACK
4410  021432  010146                            MOV     R1,-(SP)        ;;PUSH R1 ON STACK
4411  021434  010246                            MOV     R2,-(SP)        ;;PUSH R2 ON STACK
4412  021436                           1$:
4413                                    ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $RDLIN ROUTINE
4414                                    ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
4415  021436  106746                            MFPS    -(SP)           ;PUT THE PROCESSOR STATUS ON THE STACK
4416  021440  105066  000001                    CLRB    1(SP)           ;HIGH BYTE CLEARED TO INSURE KERNEL MODE
4417                                                                    ;ON PSW RETURN.
4418  021444  004767  177416                    JSR     PC,     $RDLIN  ;GO TO THE SUBROUTINE
4419  021450  012600                            MOV     (SP)+,R0        ;;GET ADDRESS OF 1ST CHARACTER
4420  021452  010067  000102                    MOV     R0,5$           ;;AND SAVE IT
4421  021456  005001                            CLR     R1              ;;CLEAR DATA WORD
4422  021460  005002                            CLR     R2
4423  021462  112046                  2$:       MOVB    (R0)+,-(SP)     ;;PICKUP THIS CHARACTER
4424  021464  001420                            BEQ     3$              ;;IF ZERO GET OUT
4425  021466  122716  000060                    CMPB    #'0,(SP)        ;;MAKE SURE THIS CHARACTER
4426  021472  003026                            BGT     4$              ;;IS AN OCTAL DIGIT
4427  021474  122716  000067                    CMPB    #'7,(SP)
4428  021500  002423                            BLT     4$
4429  021502  006301                            ASL     R1              ;;*2
4430  021504  006102                            ROL     R2
4431  021506  006301                            ASL     R1              ;;*4
```

```
                                        ROL     R2
                                        ASL     R1              ;;*8
                                        ROL     R2
                                        BIC     #^C7,(SP)       ;;STRIP THE ASCII JUNK
                                        ADD     (SP)+,R1        ;;ADD IN THIS DIGIT
                                        BR      2$              ;;LOOP
                                3$:     TST     (SP)+           ;;CLEAN TERMINATOR FROM STACK
                                        MOV     R1,12(SP)       ;;SAVE THE RESULT
                                        MOV     R2,$HIOCT
                                        MOV     (SP)+,R2        ;;POP STACK INTO R2
                                        MOV     (SP)+,R1        ;;POP STACK INTO R1
                                        MOV     (SP)+,R0        ;;POP STACK INTO R0
                                        RTI                     ;;RETURN
                                4$:     TST     (SP)+           ;;CLEAN PARTIAL FROM STACK
                                        CLRB    (R0)            ;;SET A TERMINATOR
                                        JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
                                5$:     .WORD   0
                                        JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
                                        .WORD   $QUES           ;ADDRESS OF MESSAGE TO BE TYPED
                                        BR      1$              ;;TRY AGAIN
                                $HIOCT: .WORD   0               ;;HIGH ORDER BITS GO HERE

                                ;;*****************************************************************
                                ;* SUBROUTINE TO PASS RELOCATED MESSAGE ADDRESSES TO THE $TYPE ROUTINE.
                                ;*  CALL:      JSR     R5,     $PRINT
                                ;*             <MESSAGE VIRTUAL ADDRESS>
                                ;*****************************************************************
                                $PRINT: MOV     (R5)+,  1$      ;GET THE MESSAGE VIRTUAL ADDRESS.
                                        ADD     RELOCF, 1$      ;MAKE IT PHYSICAL.
                                ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPE ROUTINE
                                ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
                                        MFPS    -(SP)           ;PUT THE PROCESSOR STATUS ON THE STACK
                                        CLRB    1(SP)           ;HIGH BYTE CLEARED TO INSURE KERNEL MODE
                                                                ;ON PSW RETURN.
                                        JSR     PC,     $TYPE   ;GO TO THE SUBROUTINE
                                1$:     .WORD   0               ;CONTAINS THE PHYSICAL MESSAGE ADDRESS.
                                        RTS     R5              ;RETURN.

                                .SBTTL  TYPE ROUTINE

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

                                $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
```

```
4488  021630  100002                        BPL     1$              ;;BR IF YES
4489  021632  000000                        HALT                    ;;HALT HERE IF NO TERMINAL
4490  021634  000430                        BR      3$              ;;LEAVE
4491  021636  010046                1$:     MOV     R0,-(SP)        ;;SAVE R0
4492  021640  017600  000002              MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
4493  021644  122767  000001  157342        CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
4494  021652  001011                        BNE     62$             ;;NO,GO CHECK FOR APT CONSOLE
4495  021654  132767  000100  157333        BITB    #APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
4496  021662  001405                        BEQ     62$             ;;NO,GO CHECK FOR CONSOLE
4497  021664  010067  000004              MOV     R0,61$          ;;SETUP MESSAGE ADDRESS FOR APT
4498  021670  004767  000274              JSR     PC,$ATY3        ;;SPOOL MESSAGE TO APT
4499  021674  000000                61$:    .WORD   0               ;;MESSAGE ADDRESS
4500  021676  132767  000040  157311  62$:  BITB    #APTCSUP,$ENVM  ;;APT CONSOLE SUPPRESSED
4501  021704  001003                        BNE     60$             ;;YES,SKIP TYPE OUT
4502  021706  112046                2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
4503  021710  001005                        BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
4504  021712  005726                        TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
4505  021714  012600                60$:    MOV     (SP)+,R0        ;;RESTORE R0
4506  021716  062716  000002          3$:   ADD     #2,(SP)         ;;ADJUST RETURN PC
4507  021722  000002                        RTI                     ;;RETURN
4508  021724  122716  000011          4$:   CMPB    #HT,(SP)        ;;BRANCH IF <HT>
4509  021730  001431                        BEQ     8$
4510  021732  122716  000200              CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
4511  021736  001007                        BNE     5$
4512  021740  005726                        TST     (SP)+           ;;POP  <CR><LF> EQUIV
4513  021742  004567  177626              JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
4514  021746  001171                        $CRLF
4515  021750  105067  000202              CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
4516  021754  000754                        BR      2$              ;;GET NEXT CHARACTER
4517  021756  004767  000056          5$:   JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
4518  021762  126726  157160          6$:   CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
4519  021766  001347                        BNE     2$              ;;IF NO GO GET NEXT CHAR.
4520  021770  016746  157150              MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
4521                                                                ;;AND THE NULL CHAR.
4522  021774  105366  000001          7$:   DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
4523  022000  002770                        BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
4524  022002  004767  000032              JSR     PC,$TYPEC       ;;GO TYPE A NULL
4525  022006  105367  000144              DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
4526  022012  000770                        BR      7$              ;;LOOP
4527
4528                                  ;HORIZONTAL TAB PROCESSOR
4529
4530  022014  112716  000040          8$:   MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
4531  022020  004767  000014          9$:   JSR     PC,$TYPEC       ;;TYPE A SPACE
4532  022024  132767  000007  000124        BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
4533  022032  001372                        BNE     9$              ;;TAB STOP
4534  022034  005726                        TST     (SP)+           ;;POP SPACE OFF STACK
4535  022036  000723                        BR      2$              ;;GET NEXT CHARACTER
4536  022040                          $TYPEC:
4537  022040  105777  157070              TSTB    @$TKS           ;;CHAR IN KYBD BUFFER?                  ;MJD001
4538  022044  100022                        BPL     10$             ;;BR IF NOT                             ;MJD001
4539  022046  017746  157064              MOV     @$TKB,-(SP)     ;;GET CHAR                              ;MJD001
4540  022052  042716  177600              BIC     #177600,(SP)    ;;STRIP EXTRANEOUS BITS                 ;MJD001
4541  022056  122716  000023              CMPB    #$XOFF,(SP)     ;;WAS CHAR XOFF                         ;MJD001
4542  022062  001012                        BNE     102$            ;;BR IF NOT                             ;MJD001
4543  022064                          101$:                                                                 ;MJD001
```

```
 4544  022064  105777  157044            TSTB    @$TKS           ;;WAIT FOR CHAR                         ;MJD001
 4545  022070  100375                    BPL     10$                                                     ;MJD001
 4546  022072  117716  157040            MOVB    @$TKB,(SP)      ;;GET CHAR                              ;MJD001
 4547  022076  042716  177600            BIC     #177600,(SP)    ;;STRIP IT                              ;MJD001
 4548  022102  122716  000021            CMPB    #$XON,(SP)      ;;WAS IT XON?                           ;MJD001
 4549  022106  001366                    BNE     10$             ;;BR IF NOT                             ;MJD001
 4550  022110                    102$:                                                                   ;MJD001
 4551  022110  005726                    TST     (SP)+           ;;FIX STACK                             ;MJD001
 4552  022112                    10$:                                                                    ;MJD001
 4553  022112  105777  157022            TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
 4554  022116  100375                    BPL     10$                                                     ;MJD001
 4555  022120  116677  000002  157014    MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
 4556  022126  122766  000015  000002    CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
 4557  022134  001003                    BNE     1$              ;;BRANCH IF NO
 4558  022136  105067  000014            CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
 4559  022142  000406                    BR      $TYPEX          ;;EXIT
 4560  022144  122766  000012  000002  1$:     CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
 4561  022152  001402                    BEQ     $TYPEX          ;;BRANCH IF YES
 4562  022154  105267                    INCB    (PC)+           ;;COUNT THE CHARACTER
 4563  022156  000000            $CHARCNT:.WORD  0               ;;CHARACTER COUNT STORAGE
 4564  022160  000207            $TYPEX: RTS     PC
 4565
 4566                            .SBTTL  APT COMMUNICATIONS ROUTINE
 4567
 4568                            ;;**************************************************************************
 4569  022162  112767  000001  000376  $ATY1:  MOVB    #1,$FFLG        ;;TO REPORT FATAL ERROR
 4570  022170  112767  000001  000366  $ATY3:  MOVB    #1,$MFLG        ;;TO TYPE A MESSAGE
 4571  022176  000403                    BR      $ATYC
 4572  022200  112767  000001  000360  $ATY4:  MOVB    #1,$FFLG        ;;TO ONLY REPORT FATAL ERROR
 4573  022206                    $ATYC:
 4574  022206  010046                    MOV     R0,-(SP)        ;;PUSH R0 ON STACK
 4575  022210  010146                    MOV     R1,-(SP)        ;;PUSH R1 ON STACK
 4576  022212  105767  000346            TSTB    $MFLG           ;;SHOULD TYPE A MESSAGE?
 4577  022216  001450                    BEQ     5$              ;;IF NOT:  BR
 4578  022220  122767  000001  156766    CMPB    #APTENV,$ENV    ;;OPERATING UNDER APT?
 4579  022226  001031                    BNE     3$              ;;IF NOT:  BR
 4580  022230  132767  000100  156757    BITB    #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
 4581  022236  001425                    BEQ     3$              ;;IF NOT:  BR
 4582  022240  017600  000004            MOV     @4(SP),R0       ;;GET MESSAGE ADDR.
 4583  022244  062766  000002  000004    ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
 4584  022252  005767  156716          1$:     TST     $MSGTYPE        ;;SEE IF DONE W/ LAST XMISSION?
 4585  022256  001375                    BNE     1$              ;;IF NOT:  WAIT
 4586  022260  010067  156724            MOV     R0,$MSGAD       ;;PUT ADDR IN MAILBOX
 4587  022264  105720                  2$:     TSTB    (R0)+           ;;FIND END OF MESSAGE
 4588  022266  001376                    BNE     2$
 4589  022270  166700  156714            SUB     $MSGAD,R0       ;;SUB START OF MESSAGE
 4590  022274  006200                    ASR     R0              ;;GET MESSAGE LNGTH IN WORDS
 4591  022276  010067  156710            MOV     R0,$MSGLGT      ;;PUT LENGTH IN MAILBOX
 4592  022302  012767  000004  156664    MOV     #4,$MSGTYPE     ;;TELL APT TO TAKE MSG.
 4593  022310  000413                    BR      5$
 4594  022312  017667  000004  000016  3$:     MOV     @4(SP),4$       ;;PUT MSG ADDR IN JSR LINKAGE
 4595  022320  062766  000002  000004    ADD     #2,4(SP)                ;;BUMP RETURN ADDRESS
 4596  022326  016746  155444            MOV     177776,-(SP)    ;;PUSH 177776 ON STACK
 4597  022332  004767  177266            JSR     PC,$TYPE        ;;CALL TYPE MACRO
 4598  022336  000000                  4$:     .WORD   0
 4599  022340                           5$:
```

```
4600  022340  105767  000221            TSTB    $LFLG           ;;SHOULD LOG AN ERROR?
4601  022344  001422                    BEQ     10$             ;;IF NOT:  BR
4602  022346  017600  000004            MOV     @4(SP),R0               ;;GET ERROR #
4603  022352  062766  000002  000004    ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
4604  022360  012701  001334            MOV     #$ASTAT,R1      ;;POINT TO TABLE START
4605  022364  005711            6$:     TST     (R1)            ;;END OF TABLE?
4606  022366  100404                    BMI     8$              ;;IF SO:  BR
4607  022370  020021                    CMP     R0,(R1)+        ;;PROPER ENTRY?
4608  022372  001406                    BEQ     9$              ;;IF SO:  BR
4609  022374  005721                    TST     (R1)+           ;;MOVE PAST COUNTER WORD
4610  022376  000772                    BR      6$              ;;KEEP LOOKING
4611  022400  026701  157076    8$:     CMP     $APTR,R1        ;;TABLE FULL?
4612  022404  001402                    BEQ     10$             ;;IF SO:  BR -- NO MORE ROOM
4613  022406  010021                    MOV     R0,(R1)+        ;;SET UP NEW ENTRY
4614  022410  005211            9$:     INC     (R1)            ;;BUMP ERROR COUNT
4615  022412  105767  000150    10$:    TSTB    $FFLG           ;;SHOULD REPORT FATAL ERROR?
4616  022416  001416                    BEQ     12$             ;;IF NOT:  BR
4617  022420  005767  156570            TST     $ENV            ;;RUNNING UNDER APT?
4618  022424  001413                    BEQ     12$             ;;IF NOT:  BR
4619  022426  005767  156542    11$:    TST     $MSGTYPE        ;;FINISHED LAST MESSAGE?
4620  022432  001375                    BNE     11$             ;;IF NOT:  WAIT
4621  022434  017667  000004  156534    MOV     @4(SP),$FATAL   ;;GET ERROR #
4622  022442  062766  000002  000004    ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
4623  022450  005267  156520            INC     $MSGTYPE        ;;TELL APT TO TAKE ERROR
4624  022454  105067  000106    12$:    CLRB    $FFLG           ;;CLEAR FATAL FLAG
4625  022460  105067  000101            CLRB    $LFLG           ;;CLEAR LOG FLAG
4626  022464  105067  000074            CLRB    $MFLG           ;;CLEAR MESSAGE FLAG
4627  022470  012601                    MOV     (SP)+,R1        ;;POP STACK INTO R1
4628  022472  012600                    MOV     (SP)+,R0        ;;POP STACK INTO R0
4629  022474  000207                    RTS     PC              ;;RETURN
4630  022476            $ATY6:
4631  022476  010046                    MOV     R0,-(SP)        ;;PUSH R0 ON STACK
4632  022500  016700  156776            MOV     $APTR,R0
4633  022504  162700  001334            SUB     #$ASTAT,R0      ;;GET SIZE OF STAT TABLE
4634  022510  005767  156460    1$:     TST     $MSGTY          ;;SEE IF DONE LAST COMMUNICATION
4635  022514  001375                    BNE     1$              ;;IF NOT:  WAIT
4636  022516  010067  156470            MOV     R0,$MSGLG       ;;SET MESSAGE LENGTH
4637  022522  012767  001334  156460    MOV     #$ASTAT,$MSGAD  ;;SET MESSAGE ADDR.
4638  022530  012767  000002  156436    MOV     #2,$MSGTY       ;;TELL APT TO TAKE STATS.
4639  022536  012600                    MOV     (SP)+,R0        ;;POP STACK INTO R0
4640  022540  000207                    RTS     PC              ;;RETURN
4641  022542            $ATY7:
4642  022542  010046                    MOV     R0,-(SP)        ;;PUSH R0 ON STACK
4643  022544  012701  001334            MOV     #$ASTAT,R1      ;;GET START OF TABLE
4644  022550  005721            1$:     TST     (R1)+           ;;END OF TABLE?
4645  022552  100402                    BMI     2$              ;;IF SO:  BR
4646  022554  005021                    CLR     (R1)+           ;;CLEAR ERROR COUNT
4647  022556  000774                    BR      1$              ;;KEEP CLEARING
4648  022560            2$:
4649  022560  012600                    MOV     (SP)+,R0        ;;POP STACK INTO R0
4650  022562  000207                    RTS     PC              ;;RETURN
4651  022564     000    $MFLG:  .BYTE   0               ;;MESSG. FLAG
4652  022565     000    $LFLG:  .BYTE   0               ;;LOG FLAG
4653  022566     000    $FFLG:  .BYTE   0               ;;FATAL FLAG
4654          022570            .EVEN
4655          000200    APTSIZE 200
```

```
 4656           000001                  APTENV=001
 4657           000100                  APTSPOOL=100
 4658           000040                  APTCSUP=040
 4659                                   ;;*****************************************************************
 4660
 4661                                   .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
 4662
 4663                                   ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
 4664                                   ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
 4665                                   ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
 4666                                   ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
 4667                                   ;*REPLACED WITH SPACES.
 4668                                   ;*CALL:
 4669                                   ;*      MOV     NUM,-(SP)       ;;PUT THE BINARY NUMBER ON THE STACK
 4670                                   ;*      TYPDS                   ;;GO TO THE ROUTINE
 4671
 4672  022570                           $TYPDS:
 4673  022570  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
 4674  022572  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
 4675  022574  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
 4676  022576  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
 4677  022600  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
 4678  022602  012746  020200                  MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
 4679  022606  016605  000020                  MOV     20(SP),R5       ;;GET THE INPUT NUMBER
 4680  022612  100004                          BPL     1$              ;;BR IF INPUT IS POS.
 4681  022614  005405                          NEG     R5              ;;MAKE THE BINARY NUMBER POS.
 4682  022616  112766  000055  000001          MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
 4683  022624  016700  155702          1$:     MOV     RELOCF, R0      ;GET RELOCATION FACTOR.
 4684  022630  012703  023012                  MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
 4685  022634  060003                          ADD     R0,     R3      ;ADD IN RELOCATION FACTOR.
 4686  022636  112723  000040                  MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
 4687  022642  005002                  2$:     CLR     R2              ;;CLEAR THE BCD NUMBER
 4688  022644  016001  023002                  MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
 4689  022650  160105                  3$:     SUB     R1,R5           ;;FORM THIS BCD DIGIT
 4690  022652  002402                          BLT     4$              ;;BR IF DONE
 4691  022654  005202                          INC     R2              ;;INCREASE THE BCD DIGIT BY 1
 4692  022656  000774                          BR      3$
 4693  022660  060105                  4$:     ADD     R1,R5           ;;ADD BACK THE CONSTANT
 4694  022662  005702                          TST     R2              ;;CHECK IF BCD DIGIT=0
 4695  022664  001002                          BNE     5$              ;;FALL THROUGH IF 0
 4696  022666  105716                          TSTB    (SP)            ;;STILL DOING LEADING 0'S?
 4697  022670  100407                          BMI     7$              ;;BR IF YES
 4698  022672  106316                  5$:     ASLB    (SP)            ;;MSD?
 4699  022674  103003                          BCC     6$              ;;BR IF NO
 4700  022676  116663  000001  177777          MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
 4701  022704  052702  000060          6$:     BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
 4702  022710  052702  000040          7$:     BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
 4703  022714  110223                          MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
 4704  022716  005720                          TST     (R0)+           ;;JUST INCREMENTING
 4705  022720  020067  157110                  CMP     R0,     .EIGHT  ;CHECK THE TABLE INDEX
 4706  022724  103746                          BLO     2$              ;GO DO THE NEXT DIGIT
 4707  022726  101002                          BHI     8$              ;;GO TO EXIT
 4708  022730  010502                          MOV     R5,R2           ;;GET THE LSD
 4709  022732  000764                          BR      6$              ;;GO CHANGE TO ASCII
 4710  022734  105726                  8$:     TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
 4711  022736  100003                          BPL     9$              ;;BR IF NO
```

```
 4712  022740  116663  177777  177776          MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
 4713  022746  105013                  9$:     CLRB    (R3)            ;;SET THE TERMINATOR
 4714  022750  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
 4715  022752  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
 4716  022754  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
 4717  022756  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
 4718  022760  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
 4719  022762  004567  176606                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 4720  022766  023012                          .WORD   $DBLK           ;ADDRESS OF MESSAGE TO BE TYPED
 4721  022770  016666  000002  000004          MOV     2(SP),4(SP)     ;;ADJUST THE STACK
 4722  022776  012616                          MOV     (SP)+,(SP)
 4723  023000  000002                          RTI                     ;;RETURN TO USER
 4724  023002  023420                  $DTBL:  10000.
 4725  023004  001750                          1000.
 4726  023006  000144                          100.
 4727  023010  000012                          10.
 4728  023012  000004                  $DBLK:  .BLKW   4
 4729                                  .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
 4730
 4731                                  ;;**************************************************************
 4732                                  ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
 4733                                  ;*OCTAL (ASCII) NUMBER AND TYPE IT.
 4734                                  ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
 4735                                  ;*CALL:
 4736                                  ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
 4737                                  ;*      TYPOS                   ;;CALL FOR TYPEOUT
 4738                                  ;*      .BYTE   N               ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
 4739                                  ;*      .BYTE   M               ;;M-1 OR 0
 4740                                  ;*                                      ;;1=TYPE LEADING ZEROS
 4741                                  ;*                                      ;;0=SUPPRESS LEADING ZEROS
 4742                                  ;*
 4743                                  ;*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
 4744                                  ;*$TYPOS OR $TYPOC
 4745                                  ;*CALL:
 4746                                  ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
 4747                                  ;*      TYPON                   ;;CALL FOR TYPEOUT
 4748                                  ;*
 4749                                  ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
 4750                                  ;*CALL:
 4751                                  ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
 4752                                  ;*      TYPOC                   ;;CALL FOR TYPEOUT
 4753
 4754  023022  017646  000000          $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
 4755  023026  116667  000001  000213          MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
 4756  023034  112667  000211                  MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
 4757  023040  062716  000002                  ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
 4758  023044  000406                          BR      $TYPON
 4759  023046  112767  000001  000173  $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
 4760  023054  112767  000006  000167          MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
 4761  023062  112767  000005  000156  $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
 4762  023070  010346                          MOV     R3,-(SP)        ;;SAVE R3
 4763  023072  010446                          MOV     R4,-(SP)        ;;SAVE R4
 4764  023074  010546                          MOV     R5,-(SP)        ;;SAVE R5
 4765  023076  116704  000147                  MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
 4766  023102  005404                          NEG     R4
 4767  023104  062704  000006                  ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
```

```
 4768  023110  110467  000134              MOVB    R4,$OMODE       ;;SAVE IT FOR USE
 4769  023114  116704  000127              MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
 4770  023120  016605  000012              MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
 4771  023124  005003                      CLR     R3              ;;CLEAR THE OUTPUT WORD
 4772  023126  006105                1$:   ROL     R5              ;;ROTATE MSB INTO "C"
 4773  023130  000404                      BR      3$              ;;GO DO MSB
 4774  023132  006105                2$:   ROL     R5              ;;FORM THIS DIGIT
 4775  023134  006105                      ROL     R5
 4776  023136  006105                      ROL     R5
 4777  023140  010503                      MOV     R5,R3
 4778  023142  006103                3$:   ROL     R3              ;;GET LSB OF THIS DIGIT
 4779  023144  105367  000100              DECB    $OMODE          ;;TYPE THIS DIGIT?
 4780  023150  100017                      BPL     7$              ;;BR IF NO
 4781  023152  042703  177770              BIC     #177770,R3      ;;GET RID OF JUNK
 4782  023156  001002                      BNE     4$              ;;TEST FOR 0
 4783  023160  005704                      TST     R4              ;;SUPPRESS THIS 0?
 4784  023162  001403                      BEQ     5$              ;;BR IF YES
 4785  023164  005204                4$:   INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
 4786  023166  052703  000060              BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
 4787  023172  052703  000040        5$:   BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
 4788  023176  110367  000042              MOVB    R3,8$           ;;SAVE FOR TYPING
 4789  023202  004567  176366              JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 4790  023206  023244                      .WORD   8$              ;ADDRESS OF MESSAGE TO BE TYPED
 4791  023210  105367  000032        7$:   DECB    $OCNT           ;;COUNT BY 1
 4792  023214  003346                      BGT     2$              ;;BR IF MORE TO DO
 4793  023216  002402                      BLT     6$              ;;BR IF DONE
 4794  023220  005204                      INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
 4795  023222  000743                      BR      2$              ;;GO DO THE LAST DIGIT
 4796  023224  012605                6$:   MOV     (SP)+,R5        ;;RESTORE R5
 4797  023226  012604                      MOV     (SP)+,R4        ;;RESTORE R4
 4798  023230  012603                      MOV     (SP)+,R3        ;;RESTORE R3
 4799  023232  016666  000002  000004      MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
 4800  023240  012616                      MOV     (SP)+,(SP)
 4801  023242  000002                      RTI                     ;;RETURN
 4802  023244     000                8$:   .BYTE   0               ;;STORAGE FOR ASCII DIGIT
 4803  023245     000                      .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
 4804  023246     000                $OCNT:.BYTE   0               ;;OCTAL DIGIT COUNTER
 4805  023247     000                $OFILL:.BYTE  0               ;;ZERO FILL SWITCH
 4806  023250  000000                $OMODE:.WORD  0               ;;NUMBER OF DIGITS TO TYPE
 4807                                ;ERROR TRAP SERVICE ROUTINE
 4808  023252  005727                ERRTRP: TST   (PC)+           ;CHECK IF PREV TRAP TO 4 REPORTED
 4809  023254  000000                1$:   .WORD   0               ;CONTAINS ERROR REPORTED FLAG
 4810  023256  001010                      BNE     2$              ;BRANCH IF NOT REPORTED
 4811  023260  005267  177770              INC     1$              ;SET DOUBLE TRAP FLAG.
 4812  023264  011667  155666              MOV     (SP),   $TMP3   ;SAVE THE BAD PC FOR TYPOUT.
 4813  023270  004767  174372              JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
 4814  023274  000031                      .WORD   31              ;ERROR TYPE CODE.
 4815  023276  000401                      BR      3$              ;SKIP HALT
 4816  023300  000000                2$:   HALT                    ;ERROR! SECOND TRAP TO 4 OCCURRED
 4817                                                              ;BEFORE FIRST WAS PRINTED
 4818  023302  005067  177746        3$:   CLR     1$
 4819  023306  000002                      RTI                     ;RETURN TO PROGRAM AND TRY TO RECOVER
 4820
 4821                                .SBTTL  PHYSICAL ADDRESS TYPE ROUTINE
 4822                                ;*  ROUTINE TO TYPE A PHYSICAL ADDRESS (22 BITS).
 4823  023310                        $TYPAD:
```

```
4824  023310  010046                        MOV     R0,-(SP)        ;;PUSH R0 ON STACK
4825  023312  010146                        MOV     R1,-(SP)        ;;PUSH R1 ON STACK
4826  023314  010246                        MOV     R2,-(SP)        ;;PUSH R2 ON STACK
4827  023316  010346                        MOV     R3,-(SP)        ;;PUSH R3 ON STACK
4828  023320  016602  000012                MOV     12(SP), R2      ;GET BASE ADDRESS
4829  023324  005003                        CLR     R3              ;WORKING & INDEX REGISTER
4830  023326  005767  155204                TST     MMAVA           ;CHECK FOR MEM MGMT AVAILABLE
4831  023332  001430                        BEQ     1$              ;BRANCH IF NO MEM MGMT
4832  023334  032737  000001  177572        BIT     #1,     @#SR0   ;CHECK IF MEM MGMT ENABLED
4833  023342  001424                        BEQ     1$              ;BRANCH IF MEM MGMT NOT ENABLED
4834  023344  010201                        MOV     R2,     R1      ;COPY VIRTUAL ADR
4835  023346  006101                        ROL     R1              ;SHUFFLE BITS 13,14,15 INTO 1,2,3
4836  023350  006101                        ROL     R1
4837  023352  006101                        ROL     R1
4838  023354  006101                        ROL     R1
4839  023356  006101                        ROL     R1
4840  023360  042701  177761                BIC     #177761, R1     ;CLR ALL EXCEPT BITS 1,2,3
4841  023364  062701  172340                ADD     #KIPAR0, R1     ;SET TO APPROPRIATE PAR
4842  023370  011101                        MOV     (R1),   R1      ;GET CONTENTS OF PAR
4843  023372  012700  000006                MOV     #6,     R0      ;SET UP COUNTER
4844  023376  006301                4$:     ASL     R1              ;SHIFT PAR
4845  023400  006103                        ROL     R3              ;SAVE OVERFLOW BITS
4846  023402  077003                        SOB     R0,     4$      ;COUNT SIX SHIFTS
4847  023404  042702  160000                BIC     #160000, R2     ;SAVE BANK BITS
4848  023410  060102                        ADD     R1,     R2      ;COMPUTE PHYSICAL ADDRESS
4849  023412  005503                        ADC     R3              ;MAKE SURE CARRY ISN'T LOST!
4850  023414  006302                1$:     ASL     R2              ;FIRST DIGIT TO R3
4851  023416  006103                        ROL     R3
4852  023420  006002                        ROR     R2              ;RESTORE R2 FOR BITS 14-0
4853  023422  010346                        MOV     R3,-(SP)        ;;SAVE R3 FOR TYPEOUT
4854                                                                ;;TYPE ADDRESS BITS 21-15
4855                                ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOS ROUTINE
4856                                ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
4857  023424  106746                        MFPS    -(SP)           ;PUT THE PROCESSOR STATUS ON THE STACK
4858  023426  105066  000001                CLRB    1(SP)           ;HIGH BYTE CLEARED TO INSURE KERNEL MODE
4859                                                                ;ON PSW RETURN.
4860  023432  004767  177364                JSR     PC,     $TYPOS  ;GO TO THE SUBROUTINE
4861  023436     003                        .BYTE   3               ;;TYPE 3 DIGIT(S)
4862  023437     001                        .BYTE   1               ;;TYPE LEADING ZEROS
4863  023440  010246                        MOV     R2,-(SP)        ;;SAVE R2 FOR TYPEOUT
4864                                                                ;;TYPE ADDRESS BITS 14-0
4865                                ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOS ROUTINE
4866                                ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
4867  023442  106746                        MFPS    -(SP)           ;PUT THE PROCESSOR STATUS ON THE STACK
4868  023444  105066  000001                CLRB    1(SP)           ;HIGH BYTE CLEARED TO INSURE KERNEL MODE
4869                                                                ;ON PSW RETURN.
4870  023450  004767  177346                JSR     PC,     $TYPOS  ;GO TO THE SUBROUTINE
4871  023454     005                        .BYTE   5               ;;TYPE 5 DIGIT(S)
4872  023455     001                        .BYTE   1               ;;TYPE LEADING ZEROS
4873  023456  012603                        MOV     (SP)+,R3        ;;POP STACK INTO R3
4874  023460  012602                        MOV     (SP)+,R2        ;;POP STACK INTO R2
4875  023462  012601                        MOV     (SP)+,R1        ;;POP STACK INTO R1
4876  023464  012600                        MOV     (SP)+,R0        ;;POP STACK INTO R0
4877  023466  012616                        MOV     (SP)+,  (SP)    ;ADJUST THE STACK TO CLEAR DATA
4878  023470  000207                        RTS     PC              ;RETURN
4879
```

```
4880                                        .SBTTL  STANDARD PROGRAM MESSAGES
4881                                    ;;**************************************************************
4882                                    ;VARIOUS MESSAGE PRINTOUTS USED THRUOUT
4883                                    ;THE PROGRAM
4884                                    ;;**************************************************************
4885  023472  005015  052113  030461  MMAMES: .ASCIZ  <15><12>'KT11 (MEMORY MANAGEMENT) AVAILABLE'
4886  023500  024040  042515  047515
4887  023506  054522  046440  047101
4888  023514  043501  046505  047105
4889  023522  024524  040440  040526
4890  023530  046111  041101  042514
4891  023536     000
4892  023537     015  031012  020062  AVAL22: .ASCIZ  <15><12>'22 BIT ADR AVAIL'<15><12>
4893  023544  044502  020124  042101
4894  023552  020122  053101  044501
4895  023560  006514  000012
4896  023564  005015  042515  047515  MEMMES: .ASCIZ  <15><12>'MEMORY MAP:'
4897  023572  054522  046440  050101
4898  023600  000072
4899  023602  005015  040520  044522  MTMAP:  .ASCIZ  <15><12>'PARITY MEMORY MAP:'
4900  023610  054524  046440  046505
4901  023616  051117  020131  040515
4902  023624  035120     000
4903  023627     015  043012  047522  FROM:   .ASCIZ  <15><12>'FROM '
4904  023634  020115     000
4905  023637     040  047524  000040  TO:     .ASCIZ  ' TO '
4906  023644  005015  047111  052523  INSUFF: .ASCIZ  <15><12>'INSUFFICIENT MEMORY...FIRST 16K NOT ALL THERE!'
4907  023652  043106  041511  042511
4908  023660  052116  046440  046505
4909  023666  051117  027131  027056
4910  023674  044506  051522  020124
4911  023702  033061  020113  047516
4912  023710  020124  046101  020114
4913  023716  044124  051105  020505
4914  023724     000
4915  023725     015  047012  020117  MTR:    .ASCIZ  <15><12>'NO PARITY REGISTERS FOUND'
4916  023732  040520  044522  054524
4917  023740  051040  043505  051511
4918  023746  042524  051522  043040
4919  023754  052517  042116     000
4920  023761     015  051012  051505  PWRMSG: .ASCIZ  <15><12>'RESTARTING AFTER A POWER FAILURE'<15><12>
4921  023766  040524  052122  047111
4922  023774  020107  043101  042524
4923  024002  020122  020101  047520
4924  024010  042527  020122  040506
4925  024016  046111  051125  006505
4926  024024  000012
4927  024026  005015  047516  050040  NOPES:  .ASCIZ  <15><12>'NO PARITY ERRORS FOUND ON MEMORY SCAN'<15><12>
4928  024034  051101  052111  020131
4929  024042  051105  047522  051522
4930  024050  043040  052517  042116
4931  024056  047440  020116  042515
4932  024064  047515  054522  051440
4933  024072  040503  006516  000012
4934  024100  005015  051120  043517  PROREL: .ASCII  <15><12>'PROGRAM NOW RESIDES BACK AT 0 TO 8K'
4935  024106  040522  020115  047516
```

```
4936  024114  020127  042522  044523
4937  024122  042504  020123  040502
4938  024130  045503  040440  020124
4939  024136  020060  047524  034040
4940  024144     113
4941  024145     015  044012  052111          .ASCIZ  <15><12>'HIT CONTINUE FOR NORMAL RUNNING'<15><12>
4942  024152  041440  047117  044524
4943  024160  052516  020105  047506
4944  024166  020122  047516  046522
4945  024174  046101  051040  047125
4946  024202  044516  043516  005015
4947  024210     000
4948  024211     015  050012  051101  MX1:    .ASCIZ  <15><12>'PARITY REGISTER AT '
4949  024216  052111  020131  042522
4950  024224  044507  052123  051105
4951  024232  040440  020124     000
4952  024237     040  047503  052116  MX2:    .ASCIZ  ' CONTROLS '
4953  024244  047522  051514  000040
4954  024252  005015  047516  046440  NOMEM:  .ASCIZ  <15><12>'NO MEMORY FOUND.'
4955  024260  046505  051117  020131
4956  024266  047506  047125  027104
4957  024274     000
4958  024275     015  005012  044412  FADMES: .ASCII  <15><12><12><12>'INPUT ALL PARAMETERS IN OCTAL.'
4959  024302  050116  052125  040440
4960  024310  046114  050040  051101
4961  024316  046501  052105  051105
4962  024324  020123  047111  047440
4963  024332  052103  046101     056
4964  024337     015  043012  051111          .ASCIZ  <15><12>'FIRST ADDRESS:  '
4965  024344  052123  040440  042104
4966  024352  042522  051523  020072
4967  024360  000040
4968  024362  005015  040514  052123  LADMES: .ASCIZ  <15><12>'LAST ADDRESS:   '
4969  024370  040440  042104  042522
4970  024376  051523  020072  020040
4971  024404     000
4972  024405     015  037412  042101  BADADR: .ASCIZ  <15><12>'?ADDRESS IN UNMAPPED BANK?'
4973  024412  051104  051505  020123
4974  024420  047111  052440  046516
4975  024426  050101  042520  020104
4976  024434  040502  045516  000077
4977  024442  005015  042523  042514  CONST:  .ASCIZ  <15><12>'SELECT CONSTANT:'
4978  024450  052103  041440  047117
4979  024456  052123  047101  035124
4980  024464     000
4981  024465     015  052412  042516  UNEXPT: .ASCIZ  <15><12>'UNEXPECTED MEMORY PARITY ERROR'
4982  024472  050130  041505  042524
4983  024500  020104  042515  047515
4984  024506  054522  050040  051101
4985  024514  052111  020131  051105
4986  024522  047522  000122
4987  024526  005015  051120  043517  PRELOC: .ASCIZ  <15><12>'PROGRAM RELOCATED TO '
4988  024534  040522  020115  042522
4989  024542  047514  040503  042524
4990  024550  020104  047524  000040
4991  024556  005015  047515  042522  MTOE:   .ASCIZ  <15><12>'MORE THAN ONE PARITY ERROR FOUND.'
```

```
 4992  024564  052040  040510  020116
 4993  024572  047117  020105  040520
 4994  024600  044522  054524  042440
 4995  024606  051122  051117  043040
 4996  024614  052517  042116  000056
 4997  024622  005015  041523  047101  SCANM:  .ASCIZ  <15><12>'SCANNING MEMORY FOR BAD PARITY.'
 4998  024630  044516  043516  046440
 4999  024636  046505  051117  020131
 5000  024644  047506  020122  040502
 5001  024652  020104  040520  044522
 5002  024660  054524  000056
 5003  024664  005015  040520  044522  PEWNC:  .ASCIZ  <15><12>'PARITY ERROR WILL NOT CLEAR.'
 5004  024672  054524  042440  051122
 5005  024700  051117  053440  046111
 5006  024706  020114  047516  020124
 5007  024714  046103  040505  027122
 5008  024722     000
 5009  024723     015  047012  020117  NOMTST: .ASCIZ  <15><12>'NO MEMORY TESTED.'
 5010  024730  042515  047515  054522
 5011  024736  052040  051505  042524
 5012  024744  027104     000
 5013  024747     015  051412  044513  SKPMES: .ASCIZ  <15><12>'SKIPPING TEST #'
 5014  024754  050120  047111  020107
 5015  024762  042524  052123  021440
 5016  024770     000
 5017  024771     377  000377          FILL2:  .ASCIZ  <377><377>
 5018
 5019                                  .SBTTL  ERROR REPORTING MESSAGES AND TABLES.
 5020                                  ;;*****************************************************************
 5021                                  ;* MESSAGE BLOCK FOR ERROR TABLE TYPEOUTS
 5022                                  ;;*****************************************************************
 5023  024774  040520  044522  054524  DM1:    .ASCIZ  'PARITY REGISTER DATA ERROR.'
 5024  025002  051040  043505  051511
 5025  025010  042524  020122  040504
 5026  025016  040524  042440  051122
 5027  025024  051117  000056
 5028  025030  042101  051104  051505  DM2.    .ASCIZ  'ADDRESS TEST ERROR(TST1-5).'
 5029  025036  020123  042524  052123
 5030  025044  042440  051122  051117
 5031  025052  052050  052123  026461
 5032  025060  024465  000056
 5033  025064  047503  051516  040524  DM4:    .ASCIZ  'CONSTANT DATA ERROR(TST6-10).'
 5034  025072  052116  042040  052101
 5035  025100  020101  051105  047522
 5036  025106  024122  051524  033124
 5037  025114  030455  024460  000056
 5038  025122  047522  040524  044524  DM5:    .ASCIZ  'ROTATING BIT ERROR(TST11-12).'
 5039  025130  043516  041040  052111
 5040  025136  042440  051122  051117
 5041  025144  052050  052123  030461
 5042  025152  030455  024462  000056
 5043  025160  047515  020123  042522  DM6:    .ASCIZ  'MOS REFRESH TEST ERROR (TST 24-25).'
 5044  025166  051106  051505  020110
 5045  025174  042524  052123  042440
 5046  025202  051122  051117  024040
 5047  025210  051524  020124  032062
```

```
5048  025216  031055  024465  000056
5049  025224  040506  040524  020114  DM7:    .ASCIZ  'FATAL ERROR HALT'
5050  025232  051105  047522  020122
5051  025240  040510  052114     000
5052  025245     115  051101  044103  DM10:   .ASCIZ  'MARCHING 1'S AND 0'S ERROR(TST 23).''
5053  025252  047111  020107  023461
5054  025260  020123  047101  020104
5055  025266  023460  020123  051105
5056  025274  047522  024122  051524
5057  025302  020124  031462  027051
5058  025310     000
5059  025311     120  051101  052111  DM11:   .ASCIZ  'PARITY MEMORY ADDRESS ERROR(TST13).'
5060  025316  020131  042515  047515
5061  025324  054522  040440  042104
5062  025332  042522  051523  042440
5063  025340  051122  051117  052050
5064  025346  052123  031461  027051
5065  025354     000
5066  025355     104  052101  047511  DM12:   .ASCIZ  'DATIO WITH WRONG PARITY DIDN'T TRAP(TST13).''
5067  025362  053440  052111  020110
5068  025370  051127  047117  020107
5069  025376  040520  044522  054524
5070  025404  042040  042111  023516
5071  025412  020124  051124  050101
5072  025420  052050  052123  031461
5073  025426  027051     000
5074  025431     127  047522  043516  DM13:   .ASCIZ  'WRONG PARITY DETECTED, BUT NO REGISTER SHOWS ERROR FLAG.'
5075  025436  050040  051101  052111
5076  025444  020131  042504  042524
5077  025452  052103  042105  020054
5078  025460  052502  020124  047516
5079  025466  051040  043505  051511
5080  025474  042524  020122  044123
5081  025502  053517  020123  051105
5082  025510  047522  020122  046106
5083  025516  043501  000056
5084  025522  040520  044522  054524  DM14:   .ASCIZ  'PARITY REGISTER NOT MAPPED AS CONTROLLING THIS ADDRESS(TST13).'
5085  025530  051040  043505  051511
5086  025536  042524  020122  047516
5087  025544  020124  040515  050120
5088  025552  042105  040440  020123
5089  025560  047503  052116  047522
5090  025566  046114  047111  020107
5091  025574  044124  051511  040440
5092  025602  042104  042522  051523
5093  025610  052050  052123  031461
5094  025616  027051     000
5095  025621     115  051117  020105  DM16:   .ASCIZ  'MORE THAN ONE REGISTER INDICATED PARITY ERROR.'
5096  025626  044124  047101  047440
5097  025634  042516  051040  043505
5098  025642  051511  042524  020122
5099  025650  047111  044504  040503
5100  025656  042524  020104  040520
5101  025664  044522  054524  042440
5102  025672  051122  051117  000056
5103  025700  040504  040524  051440  DM17:   .ASCIZ  'DATA SHOULDN'T HAVE CHANGED WHEN PARITY ERROR TRAPPED(TST13).''
```

```
5104  025706  047510  046125  047104
5105  025714  052047  044040  053101
5106  025722  020105  044103  047101
5107  025730  042507  020104  044127
5108  025736  047105  050040  051101
5109  025744  052111  020131  051105
5110  025752  047522  020122  051124
5111  025760  050101  042520  024104
5112  025766  051524  030524  024463
5113  025774  000056
5114  025776  040522  042116  046517  DM20:   .ASCIZ  'RANDOM DATA ERROR(TST14).'
5115  026004  042040  052101  020101
5116  026012  051105  047522  024122
5117  026020  051524  030524  024464
5118  026026  000056
5119  026030  047111  052123  052522  DM21:   .ASCIZ  'INSTRUCTION EXECUTION ERROR(TST15-22).'
5120  026036  052103  047511  020116
5121  026044  054105  041505  052125
5122  026052  047511  020116  051105
5123  026060  047522  024122  051524
5124  026066  030524  026465  031062
5125  026074  027051     000
5126  026077     120  047522  051107  DM23:   .ASCIZ  'PROGRAM CODE CHANGED WHEN RELOCATED.'
5127  026104  046501  041440  042117
5128  026112  020105  044103  047101
5129  026120  042507  020104  044127
5130  026126  047105  051040  046105
5131  026134  041517  052101  042105
5132  026142  000056
5133  026144  051124  050101  042520  DM24:   .ASCIZ  'TRAPPED, BUT NO REGISTER HAD ERROR BIT SET.'
5134  026152  026104  041040  052125
5135  026160  047040  020117  042522
5136  026166  044507  052123  051105
5137  026174  044040  042101  042440
5138  026202  051122  051117  041040
5139  026210  052111  051440  052105
5140  026216  000056
5141  026220  051124  050101  042520  DM25:   .ASCIZ  'TRAPPED TO 114.'
5142  026226  020104  047524  030440
5143  026234  032061  000056
5144  026240  040506  046111  042105  DM26:   .ASCIZ  'FAILED TO TRAP.'
5145  026246  052040  020117  051124
5146  026254  050101  000056
5147  026260  040450  052103  047511  DM27:   .ASCIZ  ''(ACTION ENABLE WASN'T SET).''
5148  026266  020116  047105  041101
5149  026274  042514  053440  051501
5150  026302  023516  020124  042523
5151  026310  024524  000056
5152  026314  005015  051124  050101  DM31:   .ASCIZ  <15><12>'TRAPPED TO 4 '
5153  026322  042520  020104  047524
5154  026330  032040  000040
5155
5156                                  ;;*****************************************************************
5157                                  ;DATA COLUMN HEADINGS
5158                                  ;;*****************************************************************
5159
```

```
5160  026334  041520  051011  043505  DH1:    .ASCIZ  'PC      REG       S/B      WAS'
5161  026342  020011  051440  041057
5162  026350  020011  053440  051501
5163  026356      000
5164  026357      126 050057  004503  DH2:    .ASCIZ  'V/PC    P/PC      MA         S/B      WAS'
5165  026364  027520  041520  020011
5166  026372  046440  004501  020040
5167  026400  020040  027523  004502
5168  026406  020040  020040  040527
5169  026414  000123
5170  026416  027526  041520  050011  DH12:   .ASCIZ  'V/PC    P/PC      MA         S/B'
5171  026424  050057  004503  020040
5172  026432  040515  020011  020040
5173  026440  051440  041057      000
5174  026445      126 050057  004503  DH14:   .ASCIZ  'V/PC    P/PC      REG        MA'
5175  026452  027520  041520  020011
5176  026460  051040  043505  020011
5177  026466  020040  046440  000101
5178  026474  027526  041520  050011  DH15:   .ASCIZ  'V/PC    P/PC      MAUT       REG        S/B      WAS'
5179  026502  050057  004503  020040
5180  026510  040515  052125  020011
5181  026516  020040  051040  043505
5182  026524  020040  020040  020040
5183  026532  051440  041057  020040
5184  026540  020040  053440  051501
5185  026546      000
5186  026547      126 050057  004503  DH21:   .ASCIZ  'V/PC    P/PC      IUT      MA         S/B      WAS'
5187  026554  027520  041520  020011
5188  026562  044440  052125  020011
5189  026570  046440  004501  020040
5190  026576  020040  027523  004502
5191  026604  020040  020040  040527
5192  026612  000123
5193  026614  027526  041520  050011  DH23:   .ASCIZ  'V/PC    P/PC      SRC MA     DST MA     S/B      WAS'
5194  026622  050057  004503  020040
5195  026630  051123  020103  040515
5196  026636  020040  020040  051504
5197  026644  020124  040515  020040
5198  026652  020040  027523  020102
5199  026660  020040  020040  040527
5200  026666  000123
5201  026670  027526  041520  050011  DH24:   .ASCIZ  'V/PC    P/PC      TRP/PC'
5202  026676  050057  004503  020040
5203  026704  051124  027520  041520
5204  026712      000
5205  026713      126 050057  004503  DH25:   .ASCIZ  'V/PC    P/PC      TRP/PC     REG        WAS'
5206  026720  027520  041520  020011
5207  026726  052040  050122  050057
5208  026734  020103  020040  051040
5209  026742  043505  020040  020040
5210  026750  020040  053440  051501
5211  026756      000
5212  026757      126 050057  004503  DH26:   .ASCIZ  'V/PC    P/P       REG        WAS'
5213  026764  027520  041520  020011
5214  026772  051040  043505  020011
5215  027000  020040  053440  051501
```

```
5216  027006      000
5217  027007      122  043505  020011  DH30:   .ASCIZ  'REG      WAS      MA         WAS'
5218  027014   053440  051501  020011
5219  027022   046440  004501  020040
5220  027030   020040  040527  000123
5221  027036   027526  041520  050011  DH31:   .ASCIZ  'V/PC    P/PC      MA         S/B  WAS'
5222  027044   050057  004503  020040
5223  027052   040515  020011  020040
5224  027060   051440  041057  020040
5225  027066   040527  000123
5226
5227                               ;;*****************************************************************
5228                               ;* DATA FORMAT TABLE FOR ERROR PRINTOUT.
5229                               ;;*****************************************************************
5230  027072      000      377      000  DF1:    .BYTE   0,-1,0,0
5231  027075      000
5232  027076      000      377      377  DF2:    .BYTE   0,-1,-1,0,0
5233  027101      000      000
5234  027103      000      377      377  DF3:    .BYTE   0,-1,-1,-2,-2
5235  027106      376      376
5236  027110      000      377      377  DF14:   .BYTE   0,-1,-1,-1,0,0
5237  027113      377      000      000
5238  027116      000      377      000  DF21:   .BYTE   0,-1,0,-1,0,0
5239  027121      377      000      000
5240  027124      377      000      377  DF30:   .BYTE   -1,0,-1,-2
5241  027127      376
5242                                       .EVEN
5243
5244           032000                    . - 32000                    ;THE LOADERS ARE SAVE HERE TO END OF 8K
5245                                                                  ;1536. WORDS
5246
5247           000001                          .END
```

```
ABASE = 000000            374     415
ACDW1 = 000000            374     417
ACDW2 = 000000            374     418
ACPUOP= 000000            374     389
ADDW0 = 000000            374     419
ADDW1 = 000000            374     420
ADDW10= 000000            374     429
ADDW11= 000000            374     430
ADDW12= 000000            374     431
ADDW13= 000000            374     432
ADDW14= 000000            374     433
ADDW15= 000000            374     434
ADDW2 = 000000            374     421
ADDW3 = 000000            374     422
ADDW4 = 000000            374     423
ADDW5 = 000000            374     424
ADDW6 = 000000            374     425
ADDW7 = 000000            374     426
ADDW8 = 000000            374     427
ADDW9 = 000000            374     428
ADEVCT= 000000            374     380
ADEVM = 000000            374     416
AE    - 000001            179#   2203    2289    2322    3538
AENV  = 000000            374     385
AENVM = 000000            374     386
AFATAL- 000000            374     377
AMADR1  000000            374     402
AMADR2- 000000            374     406
AMADR3  000000            374     409
AMADR4- 000000            374     412
AMAMS1  000000            374     396
AMAMS2= 000000            374     404
AMAMS3= 000000            374     407
AMAMS4= 000000            374     410
AMSGAD= 000000            374     382
AMSGLG= 000000            374     383
AMSGTY= 000000            374     376
AMTYP1= 000000            374     397
AMTYP2= 000000            374     405
AMTYP3= 000000            374     408
AMTYP4= 000000            374     411
APASS = 000000            374     379
APRIOR= 000C00            374
APTCSU= 000040           4500    4658#
APTENV= 000001           4067    4493    4578    4656#
APTSIZ= 000200           1176    4655#
APTSPO= 000100           4495    4580    4657#
ASWREG= 000000            374     387
ATESTN= 000000            374     378
AUNIT = 000000            374     381
AUSWR = 000000            374     388
AVAL22  023537           1267    4892#
AVECT1= 000000            374     413
AVECT2= 000000            374     414
BADADR  024405           1782    4972#
BANKNO  014216           1959    1968    1993    2003    3186#
```

```
BITPT   001714          577#  628  1712  1749  1762  1768  1775  2970*  3034  3037  3042*  3044  3048
                       3054  3121  3124  3129* 3131  3135  3143
BIT0  = 000001          123#  233   249  1334  1482  1488  1511  1544  1616  1662  1671  1717  1819
                       2023  2234  2267  2304  2833  2967  3054  3143  3556  3629  3704  3826
BIT00 = 000001          113#  123
BIT01 = 000002          112#  122
BIT02 = 000004          111#  121
BIT03 = 000010          110#  120
BIT04 = 000020          109#  119
BIT05 = 000040          108#  118
BIT06 = 000100          107#  117
BIT07 = 000200          106#  116
BIT08 = 000400          105#  115  3986
BIT09 = 001000          104#  114  3996  4083
BIT1  = 000002          122#
BIT10 = 002000          103#  4053
BIT11 = 004000          102#  4003
BIT12 = 010000          101#
BIT13 = 020000          100#  2264  4061
BIT14 = 040000           99#   213  2283  2289  3972
BIT15 = 100000           98#  1265  2267  2289  2937  3034  3037  3121  3124  3332
BIT2  = 000004          121#
BIT3  = 000010          120#  3048  3135  3350
BIT4  = 000020          119#  1257  1308  1518  2925  2970  3708
BIT5  = 000040          118#
BIT6  = 000100          117#
BIT7  = 000200          116#
BIT8  = 000400          115#
BIT9  = 001000          114#
BNKTAB  017350         3801*  3814  3946#
BPTVEC- 000014          130#
CHGG1   003432         1007#
CHGG2   015662         3547#
CKPMER  015662         2096  2186  3067  3548#
CLRPAR  016514         1467  1479  1538  2225  3622  3741#
CONST   024442         1787  4977#
CR    - 000015           38#  4556  4566
CRLF  - 000200           39#  1208  4510  4566
CTRLS   005734         1602  1614#
DDISP = 177570           45#   351  1164
DF1     027072         1012  5230#
DF14    027110         1069  1074  1079  1105  1110  1115  5236#
DF2     027076         1018  1028  1033  1038  1048  1090  1120  1125  5232#
DF21    027116         1095  5238#
DF3     027103         1023  1053  1058  1063  1085  5234#
DF30    027124         1130  1135  5240#
DH1     026334         1010  5160#
DH12    026416         1056  1061  5170#
DH14    026445         1066  1077  5174#
DH15    026474         1072  5178#
DH2     026357         1015  1026  1031  1036  1046  1088  5164#
DH21    026547         1093  5186#
DH23    026614         1103  5193#
DH24    026670         1108  5201#
DH25    026713         1113  5205#
DH26    026757         1118  1123  5212#
```

```
DH30    027007          1128    5217#
DH31    027036          1021    1051    1083    5221#
DIDBH   011354          2502#
DIDBL   011270          2457#
DIDO    011206          2412#
DIPDO   011444          2550#
DISMAP  005034          1310    1373    1384#
DISPLA  001132           351#   1164*   1172*   1181*   3377*   3432*   4018*   4052*
DISPRE  000174           195#   1172
DM1     024774          1009    1071    5023#
DM10    025245          1045    5052#
DM11    025311          1050    5059#
DM12    025355          1055    5066#
DM13    025431          1060    5074#
DM14    025522          1065    5084#
DM16    025621          1076    5095#
DM17    025700          1081    5103#
DM2     025030          1014    1020    5028#
DM20    025776          1087    5114#
DM21    026030          1092    5119#
DM23    026077          1102    5126#
DM24    026144          1107    5133#
DM25    026220          1112    5141#
DM26    026240          1117    5144#
DM27    026260          1122    5147#
DM31    026314          1132    5152#
DM4     025064          1025    5033#
DM5     025122          1030    5038#
DM6     025160          1035    5043#
DM7     025224          1040    5049#
DONE    012340          2819#
DPDBH   011614          2640#
DPDBL   011530          2595#
DSWR  = 177570            44#    350    1163
DT1     002040           641#   1011
DT12    002066           645#   1057    1062
DT14    002100           647#   1067    1078
DT15    002112           649#   1073
DT2     002052           643#   1017    1022    1027    1032    1037    1047    1052    1084    1089
DT21    002130           652#   1094
DT23    002146           655#   1104
DT24    002164           658#   1109
DT25    002174           660#   1114
DT26    002210           662#   1119    1124
DT30    002222           664#   1129
DT31    002234           666#   1134
EMTVEC= 000030           133#
ENDBKT  017352          3802*   3803*   3807    3930    3947#
ENDINS  017650          4020#
ERRTRP  023252           202    1385    1436    3472    4808#
ERRVEC= 000004           126#    201    1161    1162*   1173*   1244*   1254*   1287*   1332*   1385*   1419*   1436*   3373*
                        3428*   3468*   3472*   3977    3978*   3980*   3983*
FADMES  024275          1694    4958#
FILL2   024771          5017#
FLAGBK  001757           602#
FLG30K  001774           612#   1301*   3051    3139    3673    3843
```

```
FROM    023627          3863    4903#
FSTADR  001762           605#   1688*   1798*   3056    3057    3145    3531
GMPR    005136          1401    1415#
GMPRA   005164          1420#   1434
GMPRB   005200          1419    1426#
GMPRBA  005210          1424    1428#
GMPRC   005214          1429#   1430
GMPRD   005252          1417    1442#
GNS     =  ****** U      194    1207
HT      =  000011         36#   4508    4566
INITDN  013076          1853    1890    1918    1992    2002    2957#
INITEX  013204          2946    2971    2980#
INITMM  013000          1836    1879    1930    1958    1967    2053    2065    2099    2118    2139    2172    2357    2415
                        2460    2505    2553    2598    2643    2686    2763    2775    2793    2805    2934#   3218
INSUFF  023644          1403    4906#
IOTVEC= 000020           131#
KIPAR0= 172340           166#    240     256*   2915*   3167    3345*   3406*   4841
KIPAR1= 172342           167#    259*   2916*   3346*   3407*
KIPAR2= 172344           168#   1256*   1330*   1358*   1506*   2263    2280    2285    2917*   2945*   2968*   3007*   3026
                        3090*   3112    3338*   3339*   3345    3401*   3633    3635*   3712*   3719*
KIPAR3= 172346           169#   1331*   1348    1353    1359*   2918*   3026*   3027*   3112*   3113*   3331*   3333*   3338
                        3346    3402*   3634    3636*   3687    3691    3713*   3718*
KIPAR4= 172350           170#   2919*
KIPAR5= 172352           171#   2920*
KIPAR6= 172354           172#   2921*
KIPAR7= 172356           173#   2922*
KIPDR0= 172300           155#    235    2907*
KIPDR1= 172302           156#   2908*
KIPDR2= 172304           157#   2909*
KIPDR3= 172306           158#   2910*
KIPDR4= 172310           159#   2911*
KIPDR5= 172312           160#   2912*
KIPDR6= 172314           161#   2913*
KIPDR7= 172316           162#   2914*
KTSIZ   004634          1262    1269    1327#
LADMES  024362          1729    4968#
LDDISP  001512           498#   1180*
LF      =  000012         37#   4560    4566
LMAD    001510           497#   3443    3457*   3464    3478*
LMEMHI  017346          3820*   3861    3868*   3870    3912*   3922*   3924*   3945#
LMEMLO  017344          3819*   3860    3867*   3869*   3880    3913*   3921*   3923*   3944#
LSIFLG  001776           613#   1232*   1238*   2212
MAMF    015562          1816    2349    3523#
MAMF1   015646          3538#   3540
MAMF2   015660          3524    3526    3532    3542#
MANUAL  006224          1443    1603    1670    1688#   1784
MANUL1  006242          1690    1692#   1704    1726
MANUL2  006644          1691    1796#
MAPRB   005306          1470    1474#
MASK4K= 017777           183#
MASK8K= 037777           184#   1289    1304    1337    3694
MEMMAP  001514           500#    624    1226    1284    1328    1370    1384    1393    1398    1521    1723    1769    1776
                        3336    3355
MEMMES  023564          1387    4896#
MFPT    =  000007        185#   1234
MMAMES  023472          1249    4885#
```

```
MMAVA   000536   231   278#  1241* 1247* 1265* 1346  1361  1469  1504  1553  1817  2260  2278
                 2923  2939  2961  2997  3032 3084  3119  3156  3329  3399  3449  3458  3631
                 3671  3685  3706  3840  4830
MMDOWN  013556   1866  1904  1926  1998  2014 2975  3077#
MMINIT  012636   1246  1472  1821  2906#
MMORE   001754    598# 2946* 2947* 2971* 2972* 2980* 3062  3149
MMJP    013212   1849  1886  1944  1963  1978 2056  2075  2109  2130  2151  2348  2382  2429
                 2474  2520  2567  2612  2658 2745  2768  2787  2798  2817  2950  2990#  3221
MMVEC = 000250    144#
MPAMEM  005260   1438  1467#
MPEND   002266    685#
MPPATS  002242    633   674#
MPRX    003370    631   989#  1415  1432  1437 1500  1593  1601  1650  1660  1669  2165  3065
                 3523  3548
MPR0    002270    630   700#  1418  1484  1542 1614  1659  1668
MPR1    002334    718#
MPR10   003040    880#
MPR11   003104    898#
MPR12   003150    916#
MPR13   003214    934#
MPR14   003260    952#
MPR15   003324    970#
MPR2    002400    736#
MPR3    002444    754#
MPR4    002510    772#
MPR5    002554    790#
MPR6    002620    808#
MPR7    002664    826#
MPR8    002730    844#
MPR9    002774    862#
MTMAP   023602   1540  4899#
MTOE    024556   3506  4991#
MTR     023725   1440  4915#
MX1     024211   1549  4948#
MX2     024237   1586  4952#
NOMEM   024252   3810  4954#
NOMTST  024723   5009#
NONKT   004522   1243  1244  1283#
NOPES   024026   3724  4927#
OFFLG   001760    603# 2198* 2341*
PARMAT  015770   2245  3580#
PARVEC- 000114    180# 2217* 2221* 2231* 3375* 3430* 3534* 3535*
PBTRP   010356    632  2231#
PBTRP1  010366   2215  2233#
PESRV   015454    634  3486#
PEWNC   024664   3664  5003#
PHYADR  014120   1838  1856  1881  1893  1921 1932  3156#
PIRQ  = 177772    43#
PIRQVE= 000240    137#
PLUS1   002000    614# 2937* 2967* 3004  3010* 3041* 3094*
PMEMAP  001654    558#  627  1474  1497*
PRELOC  024526   3307  4987#
PRGMAP  000534    228   251   266   277#  1213* 1813  2023  2853  3320  3340  3357  3584*  3393
                 3435*
PROREL  024100   4934#
PR0   - 000000     60#
```

```
PR1     = 000040            61#
PR2     = 000100            62#
PR3     = 000140            63#
PR4     = 000200            64#
PR5     = 000240            65#
PR6     = 000300            66#
PR7     = 000340            67#
PS      = 177776            40#   41
PSCAN     016034          3512    3562    3567    3613#
PSW     = 177776            41#
PWRMSG    023761           316    4920#
PWRVEC  = 000024           132#    285*    286*    295*    301*    313*    314*   1156*   1157*   3374*   3429*
RADTAB    002002           622#   3378    3420
RANTST    011070          2355#
RELOC     014450          3282#   3342    3365    3403    3413
RELOCF    000532           261    276#   1214*   1815    2947    2972    3368*   3372*   3379    3387    3411    3434*   3527
                          4123    4144    4146    4460    4683
RELTOP    014572          2836    3320#
RELO      015106           268    2839    3393#
RESCHK    006106          1659#
RESLDR    015306          2842    3443#
RESRVD    001506           495#    629    1618*   1623    1628    1633    1642   2268*   2269    2270    2320
RESTAR    000300           199    224#    318    1212
RESTOR    000304           226#
REST1     000306           225    227#
REST2     000324           229    231#
RESVEC  = 000010           127#   1233*
ROTATE    014322          2120    2141    3227#
RW      = 000006           176#   2907    2908    2909    2910    2914
SAVLDR    015374          1222    2890    3464#
SAVTST    001614           538#    625    1392    1750    2029
SCANM     024622          3620    4997#
SECT2     007466          2023#
SELECT    003750          1146#
SELFLG    001756           601#   1144*   1146*   1689
SETAE     015626          2185    3528    3530    3534#
SETCON    014304          2117    2138    2171    3217#
S%PMFS    024747          5013#
SPREND    016646          3766    3781#
SPRNT     016536          1625    1644    1680    2194    3758#
SPRNTA    016636          3770    3774    3779#
SPRNTB    016642          3760    3780#
SPRNTP    016574          2250    2274    2294    2308    2325    3768#
SPRNTQ    016550          3509    3558    3564    3655    3762#
SPRNTO    016600          1636    1860    1898    2009    2222    2241    2334    3769#
SPRNT1    016606          1973    3772#
SPRNT2    016624          1843    1938    2070    2104    2125    2146    2373    2697    2706    2728    2781    2811    3777#
SPRNT3    016620          2423    2468    2514    2561    2606    2652    3776#
SR0     = 177572           148#    233     249*   1245*   1819   2926*   3451*   3460*   4832
SR1     = 177574           149#
SR2     = 177576           150#
SR3     = 172516           151#   1257*   1261*   2925*
STACK   = 001100            31#   1154    1283    1327    1780    1785
START     003742           198    1144#
STARTA    003756           230    1145    1147#
START1    006662           271    1797    1807#   1814    2837    2892
```

```
STKLMT- 177774    42#
SWR     001130   293#   306*   350#  1152  1163*  1165  1171*  1178*  1195  1242  1416  1796  2079
                2081   2167   2823  3376*  3431*  3525  3529   3550   3972  3986  3988  3996  4003
                4053   4061   4074  4083   4205   4252*
SWREG   000176   196#   1171   1195  4205   4221
SW0   = 000001    95#
SW00  = 000001    85#     95
SW01  = 000002    84#     94
SW02  = 000004    83#     93
SW03  = 000010    82#     92
SW04  = 000020    81#     91
SW05  = 000040    80#     90   1796
SW06  = 000100    79#     89   1416   2167
SW07  = 000200    78#     88   2823
SW08  = 000400    77#     87   2079
SW09  = 001000    76#     86
SW1   = 000002    94#
SW10  = 002000    75#
SW11  = 004000    74#
SW12  = 010000    73#   1242
SW13  = 020000    72#
SW14  = 040000    71#
SW15  = 100000    70#
SW2   = 000004    93#
SW3   = 000010    92#
SW4   = 000020    91#
SW5   = 000040    90#   3529
SW6   = 000100    89#   3525   3550
SW7   = 000200    88#
SW8   = 000400    87#
SW9   = 001000    86#
TBITVE= 000014   128#
TEMP    001770   610#  2688*  2738  3644*  3764  3821*  3916  3934*
TEMP1   001772   611#  2360*  2368  2687*  2693  2719   2733
TKVEC = 000060   135#
TMAP    005530  1526   1538#
TO      023637  3891   4905#
TPVEC = 000064   136#
TRAPVE= 000034   134#
TRTVEC= 000014   129#
TSTMAP  001554   519#   626  1813*  2031*  3044  3131
TST1    006756  1818   1820  1833#
TST10   007660  2080   2084  2093#
TST11   007736  2098   2114#
TST12   010016  2135#
TST13   010074  2163#
TST14   011064  2169   2353#
TST15   011202  2410#
TST16   011264  2455#
TST17   011350  2500#
TST2    007100  1876#
TST20   011440  2548#
TST21   011524  2593#
TST22   011610  2638#
TST23   011676  2684#
TST24   012130  2761#
```

```
TST25   012234           2791#
TST3    007200           1915#
TST32   012344            637    2821#
TST4    007306           1955#
TST5    007374           1989#
TST6    007526           2050#   2086
TST6A   007532           2052#   2087
TST7    007552           2062#   2086
TYPMAP  016650           1389    1590    3801#
UNEXPT  024465           2486    4981#
UP    = 000000            175#   2907    2908    2909    2910    2914
WWP     001766            609#   1490    1493    2202    2209
WWPBYT  010136           2176#
WWPB0   010100           2165#
WWPB1   010176           2187#   2347
WWPB2   010242           2200#   2343
WWPB3   010730           2303    2312#
WWPB4   011020           2226    2244    2338#
WWPB5   011044           2345#
WWPB6   011054           2184    2346    2348#
W3X9    014370           3250#
$APTHD  001320            446     452#
$APTR   001502            489#   4611    4632
$ASTAT  001334            462#    489    4600    4604    4630    4633    4637    4643
$ASTEN  001500            488#
$ATYC   022206           4571    4573#
$ATY1   022162           4569#
$ATY3   022170           4498    4570#
$ATY4   022200           4070    4572#
$ATY6   022476           4630#
$ATY7   022542           4641#
$AUTOB  001124            347#   1204*   4213    4393
$BASE   001250            415#
$BDADR  001112            342#    655     658     660    3298*   3486*
$BDDAT  001116            344#    641     643     649     652     655     664    3296*   3764*   3780*
$BELL   001164            366#   4056    4095
$CDW1   001254            417#
$CDW2   001256            418#
$CHARC  022156           4515*   4525*   4532    4558*   4563#
$CKSWR  020430           2996    3083    3970    4027    4048    4082    4205#
$CMTAG  001070            227     262     264     330#    623    1149    1150    1158
$CM3  = 000000            360#
$CM4  = 000004            360#    361#    362#    363#    364#
$CNTLG  021371           4217    4388#
$CNTLU  021364           4243    4356    4387#
$CPUOP  001222            389#
$CRLF   001171            368#   1391    2845    4065    4095    4107    4134    4140    4191    4256    4362    4387    4453
                         4514    4566
$DBLK   023012           4684    4720    4728#
$DDW0   001260            419#
$DDW1   001262            420#
$DDW10  001304            429#
$DDW11  001306            430#
$DDW12  001310            431#
$DDW13  001312            432#
$DDW14  001314            433#
```

```
$DDW15  001316  434#
$DDW2   001264  421#
$DDW3   001266  422#
$DDW4   001270  423#
$DDW5   001272  424#
$DDW6   001274  425#
$DDW7   001276  426#
$DDW8   001300  427#
$DDW9   001302  428#
$DEVCT  001204  380#
$DEVM   001252  416#
$DOAGN  012612  2862  2880  2887  2889  2891#
$DTBL   023002  4688  4724#
$ENDAD  012560  211  1186  2882#  4090
$ENDCT  012512  1158  2864#
$ENDMG  012616  2867  2893#
$ENULL  012633  2876  2896#
$ENV    001214  385#  1193  1215  2825  2888  4067  4493  4578  4617
$ENVM   001215  386#  1176  4495  4500  4580
$EOP    012462  2824  2855#
$EOPCT  012504  1158*  2861#  2865
$ERFLG  001073  333#  3955  3992  3994  4000*  4022  4050*  4095
$ERMAX  001105  339#  3994  4017*  4022
$ERROR  017666  1626  1637  1645  1681  1844  1861  1899  1939  1974  2010  2071  2105  2126
                2147  2195  2223  2242  2251  2275  2295  2309  2326  2335  2374  2424  2469
                2515  2562  2607  2653  2698  2707  2729  2782  2812  2951  2976  3199  3299
                3322  3360  3395  3445  3497  3510  3559  3565  3656  4042#  4813
$ERRPC  001106  340#  641  643  645  647  649  652  655  658  660  662  4058*  4059*
                4060  4095  4113  4122
$ERRTB  003432  635  1006#
$ERRTY  020116  4063  4105#
$ERTTL  001102  337#  3651*  4057*  4095
$ESCAP  001162  365#  4016*  4086  4088  4095
$ETABL  001214  384#
$ETEND  001320  435#  458
$FATAL  001176  377#  4621*
$FFLG   022566  4569*  4572*  4615  4624*  4653#
$FILLC  001146  358#  4518  4566
$FILLS  001145  357#  4566
$GDADR  001110  341#  641  643  645  647  649  652  655  664  3297*  3758*  3765*  3769*
                3772*  3773*  3777*  3778*
$GDDAT  001114  343#  641  643  645  649  652  655  3295*  3759*  3779*
$GET42  012550  2877#
$GTSWR  020502  1202  4218#
$HD  =  000000  12
$HIBTS  001320  453#
$HIOCT  021572  1705  1707*  1709*  1711*  1713  1724  1739  1742*  1743  1753  4440*  4452#
$ICNT   001074  334#  4007*  4008  4010*  4021
$ILLUP  000706  285  301  320#
$INTAG  001125  348#  4257  4393
$ITEMB  001104  338#  4060*  4069  4095  4110
$LF     001172  369#  4095  4377  4387  4453  4566
$LFLG   022565  4600  4625*  4652#
$LPADR  001076  335#  1814*  1815*  2086*  3998*  4014*  4019  4021
$LPERR  001100  336#  3998  4015*  4021  4085
$MADR1  001226  402#
```

```
$MADR2  001232  406#
$MADR3  001236  409#
$MADR4  001242  412#
$MAIL   001174  375#  454   458   1175  1193  4013  4067  4493
$MAMS1  001224  396#
$MAMS2  001230  404#
$MAMS3  001234  407#
$MAMS4  001240  410#
$MBADR  001322  454#
$MFLG   022564  4570* 4576  4626* 4651#
$MNEW   021407  4229  4391#
$MSGAD  001210  382#  4586* 4589  4637*
$MSGLG  001212  383#  4591* 4636*
$MSGTY  001174  376#  4584  4592* 4619  4623* 4634  4638*
$MSWR   021376  4220  4389#
$MTYP1  001225  397#
$MTYP2  001231  405#
$MTYP3  001235  408#
$MTYP4  001241  411#
$MXCNT  017652  4011  4021#
$NULL   001144  356#  4520  4566
$NWTST= 000001  1824# 1826  1867# 1869  1906# 1908  1946# 1948  1980# 1982  2040# 2042  2043
                2058# 2060  2089# 2091  2111# 2132# 2153# 2155  2350# 2385# 2387  2430# 2432
                2475# 2477  2521# 2523  2568# 2570  2613# 2615  2660# 2662  2748# 2750  2788#
$OCNT   023246  4761* 4791* 4804#
$OMODE  023250  4756* 4760* 4765  4768* 4779* 4806#
$OVER   017636  3973  3991  3999  4009  4018#
$PASS   001202  379#  1175* 2827  2831  2858* 2859* 2868  2893  4005  4022
$PASTM  001326  456#
$PRINT  021574  315   1188  1248  1266  1386  1390  1402  1439  1539  1548  1585  1693  1728
                1781  1786  2844  2866  2875  3306  3487  3505  3619  3663  3723  3809  3862
                3890  4055  4064  4106  4131  4133  4137  4139  4184  4190  4216  4219  4228
                4242  4255  4276  4337  4344  4350  4355  4361  4363  4367  4371  4376  4447
                4449  4459# 4513  4719  4789
$PWRAD  000702  318#
$PWRDN  000540  285#  313   1156
$PWRMG  000676  316#
$PWRUP  000612  295   301#
$QUES   001170  367#  4095  4277  4368  4387  4450  4453  4566
$RDCHR  020736  4290# 4330
$RDLIN  021066  4320# 4418
$RDOCT  021420  1702  1736  1794  4407#
$RDSZ = 000010  4313#
$SAVR6  000712  294*  302   303*  304*  322#
$SCOPE  017354  1834  1877  1916  1956  1990  2051  2063  2094  2115  2136  2164  2354  2411
                2456  2501  2549  2594  2639  2685  2762  2792  2820  3964#
$SETUP= 000130  188#  1155  1156  1158  1159  1186  1190  2857  3965  4043  4077  4090  4200
                4393
$STUP = 177777  188#
$SVLAD  017602  3981  4012#
$SVPC = 000010  209#  214
$SWR  = 167400  1#    12    17    18    19    20    21    22    23    24    319   364   365
                366   1159  1835  1878  1917  1957  1991  2052  2064  2095  2116  2137  2165
                2355  2412  2457  2502  2550  2595  2640  2686  2763  2793  2852  2857  2878
                2892  2893  3956  3957  3958  3959  3960  3972  3984  3986  3987  3992  3993
                3994  4001  4002  4003  4015  4018  4021  4034  4035  4036  4037  4038  4053
```

```
                    4061    4074    4083    4095
$SWRFG  001216       387#    1178
$SWRMK= 000340         1#      24      25    3960    3961    3988    3989
$TESTN  001200       378#    4013*
$TIMES  001160       269*     364#    2821*   2857*   4001*   4008   4011*   4021
$TKB    001136       353#    4198    4209    4235    4294    4300    4539    4546    4566
$TKS    001134       352#    4198    4207    4232    4259*   4292    4298    4537    4544    4566
$TMP0   001150       360#     647     649     652     660     662     664    3762*   3768*   3776*
$TMP1   001152       361#     660     662     664    3763*
$TMP2   001154       362#    1741
$TMP3   001156       363#     666    1742    4812*
$TN     = 000026       1#      12    1824    1835#   1867    1878#   1906    1917#   1946    1957#   1980    1991#   2040
                    2052#   2058    2060    2064#   2076    2089    2095#   2098    2111    2116#   2132    2137#   2153
                    2165#   2350    2355#   2385    2412#   2430    2457#   2475    2502#   2521    2550#   2568    2595#
                    2613    2640#   2660    2686#   2748    2763#   2788    2793#
$TPB    001142       355#    4555*   4566
$TPFLG  001147       359#    4487    4566
$TPS    001140       354#    4553    4566
$TSTM   001324       455#
$TSTNM  001072       270*     332#   2085*   2822*   3955    3990   4012*   4013    4018    4022    4052    4095
$TTYIN  021354      4322    4323    4341    4364    4382   4386#
$TYPAD  023310      3310    4169   4823#
$TYPDS  022570      2874    4164   4672#
$TYPE   021624      4466    4487#   4597
$TYPEC  022040      4261    4517    4524    4531   4536#
$TYPEX  022160      4559    4561   4564#
$TYPOC  023046      4120    4155    4227   4759#
$TYPON  023062      4758    4761#
$TYPOS  023022      1572    1582    3877    3887    3899    3909    4179   4754#   4860    4870
$UNIT   001206       381#
$UNITM  001330       457#
$USWR   001220       388#
$VECT1  001244       413#
$VECT2  001246       414#
$VERPC  001504       494#     643     645     647     649     652     655     658     660     662   4122*   4123*
$XOFF = 000023      4541    4566
$XON  = 000021      4305    4548    4566
$XTSTR  017400      3975#
$OFILL  023247      4755*   4759*   4769   4805#
$40CAT= ****** U    3972    4063
.       = 032000     190#    194#    201#    209     210#    212#    214#    215#    219#    297     321     329#    370
                      442     443#    445#    447#    604#    989#   1153    1208#   2893    2897   4021    4022    4095
                    4194#   4198    4386#   4387    4393    4453    4566   4654#   4728#   5244#
.BITPT  002014       628#    2177    2942    2964    3008    3019    3091   3103    3191    3585
.CONST  001764       608#   1795*    2052    2064
.FIGHT  002034       636#    4705
.ERRTB  002032       635#    4128
.MEMMA  002004       624#
.MPPAT  002026       633#    2095
.MPRX   002022       631#    2201    2208    2239    2302    3492    3537    3553    3645    3743
.MPRO   002020       630#    2233
.PBTRP  002024       632#    2217
.PESRV  002030       634#    2221    2231    3534
.PMEMA  002012       627#    2178    3630    3700
.RESRV  002016       629#
.SAVTS  002006       625#    1809
```

```
.STACK  002002          623#   1807
.TSTMA  002010          626#   1810    2026    3020    3105
.TST32  002036          637#
.$ASTA= ****** U       4570   4573
.$X     - 001320        442#    447
```

```
ABORT      1#  2954  2979  3202  3301  3325  3363  3398  3448
CKSWR      1#  2991  3078  3965  4022  4043  4077
CKWD       1#  1634  1841  1858  1936  2068  2102  2123  2144  2272  2292  2323  2371  2421  2466
        2512  2559  2604  2650  2695  2704  2726  2779  2809
CKWD2      1#  1840  1934  2066  2101  2370  2777  2807
COMMEN     1#   138#  1138  1801
ENDCOM     1#    12   138#  1142  1805
ERROR     32#
ESCAPE     1#  138#
GETPRI     1#  138#
GETSWR     1#  138#  1190#
GTSWR      1#  1197
LDPDR      1#  2906  2908  2909  2910  2914
MORETA   323#   437
MULT       1#  138#
NEWTST     1#  138#  1824  1867  1906  1946  1980  2040  2058  2089  2111  2132  2153  2350  2385
        2430  2475  2521  2568  2613  2660  2748  2788
POP        1#  138#   306   307  2298  2318  2331  2948  2973  3014  3028  3098  3115  3177  3207
        3311  3385  3427  3514  3541  3570  3593  3717  3726  3748  3912  3936  4441  4627  4628
        4639  4648  4714  4873
PRINT      1#  1248  1266  1386  1390  1402  1439  1539  1547  1585  1692  1727  1781  1786  2843
        3306  3487  3505  3619  3663  3723  3809
PUSH       1#  138#   287   293  2253  2282  2300  2934  2957  2999  3086  3158  3187  3282  3326
        3419  3490  3536  3552  3580  3613  3623  3633  3741  3815  3857  4409  4573  4575  4596
        4630  4641  4672  4823
RDCHR      1#  4325
RDDEC      1#
RDLIN      1#  4412
RDOCT      1#  1697  1731  1789
REPORT     1#   138#
RESREG     1#
SAVREG     1#
SCOPE     33#
SCOPEX  3950#  4020
SCOPIN  3950#  3971
SETPRI     1#   138#
SETUP      1#  138#  1147
SIMTRP     1#  1197  1567  1577  1697  1731  1789  2869  2991  3078  3872  3882  3894  3904  3965
        4022  4043  4077  4115  4150  4159  4174  4222  4325  4413  4461  4855  4865
SKIP       1#  138#  2098
SLASH      1#  138#   688   699
SPACE   138#
STARS      1#   138#   207   220   223   283   299   325   370   373   439   441   448   461   491
         493   617   621   638   640   670   672  1272  1282  1313  1326  1376  1383  1410  1413
        1446  1466  1529  1537  1608  1612  1653  1657  1824  1832  1867  1875  1906  1914  1946
        1954  1980  1988  2036  2039  2040  2049  2058  2061  2089  2092  2111  2113  2132  2134
        2153  2162  2350  2352  2385  2409  2430  2454  2475  2499  2521  2547  2568  2592  2613
        2637  2660  2683  2748  2760  2788  2790  2846  2899  2905  2930  2932  2983  2988  3070
        3075  3153  3155  3183  3185  3214  3216  3224  3226  3247  3249  3279  3281  3315  3319
        3390  3392  3438  3442  3481  3485  3519  3522  3544  3546  3573  3578  3600  3612  3738
        3740  3754  3757  3784  3800  3952  4030  4095  4197  4200  4282  4313  4395  4454  4458
        4472  4568  4659  4731  4881  4884  5020  5022  5156  5158  5227  5229
SWRSU      1#  138#  1159#
SWROR3     1#  3274
S1REGS  1824#  1826  1869  1908  1948  1982  2043
TYPADR     1#  3309  4168
```

```
TYPBIN      1#    138#
TYPBN       1#
TYPBYT      1#   4171
TYPDEC      1#    138#  2868  4158
TYPDS       1#   2869  4159
TYPE        1#    315  1188  1248  1266  1386  1390  1402  1439  1539  1548  1585  1693  1728  1781
           1786  2844  2866  2875  3306  3487  3505  3619  3663  3723  3809  3862  3890  4055  4064
           4106  4131  4133  4137  4139  4184  4190  4216  4218  4228  4242  4254  4275  4337  4344
           4350  4355  4361  4363  4366  4371  4376  4447  4449  4513  4719  4789
TYPNAM      1#    138#  1182
TYPNUM      1#    138#
TYPOC       1#   4115  4150  4222
TYPOCS      1#    138#  1564  1575  3870  3880  3892  3902  4853  4863
TYPOCT      1#    138#  4113  4149  4221
TYPON       1#
TYPOS       1#   1567  1577  3872  3882  3894  3904  4174  4855  4865
TYPTXT      1#    138#
$CKWD       1#   1635  1679  1842  1859  1897  1937  1972  2008  2069  2103  2124  2145  2193  2273
           2293  2324  2333  2372  2422  2467  2513  2560  2605  2651  2696  2705  2727  2780  2810
$INDN       1#   1853  1890  1918  1992  2002
$INMM       1#   1836  1879  1930  1958  1967  2053  2065  2099  2118  2139  2172  2357  2415  2460
           2505  2553  2598  2643  2686  2763  2775  2793  2805  3217
$MMDN       1#   1865  1903  1925  1997  2013
$MMUP       1#   1848  1885  1943  1962  1977  2055  2074  2108  2129  2150  2380  2428  2473  2519
           2566  2611  2657  2767  2786  2797  2816  3220
$$CKWD      1#   1625  1636  1644  1680  1843  1860  1898  1938  1973  2009  2070  2104  2125  2146
           2194  2222  2241  2250  2274  2294  2308  2325  2334  2373  2423  2468  2514  2561  2606
           2652  2697  2706  2728  2781  2811  3508  3558  3563  3654
$$CMRE    323#
$$CMTM    323#   360   361   362   363
$$ESCA      1#    138#
$$NEWT      1#    138#  1824  1867  1906  1946  1980  2040  2058  2089  2111  2132  2153  2350  2385
           2430  2475  2521  2568  2613  2660  2748  2788
$$SETM   1175#
$$SKIP      1#    138#  2098
.CHBHD   2748#  2750
.C11     2153#  2155
.DIDBH   2475#  2477
.DIDBL   2430#  2432
.DIDO    2385#  2387
.DIPDO   2521#  2523
.DPDBH   2613#  2615
.DPDBL   2568#  2570
.EQUAT      1#     28
.ERROR      1#   1626  1637  1645  1681  1844  1861  1899  1939  1974  2010  2071  2105  2126  2147
           2195  2223  2242  2251  2275  2295  2309  2326  2335  2374  2424  2469  2515  2562  2607
           2653  2698  2707  2729  2782  2812  2951  2976  3199  3299  3322  3359  3395  3445  3497
           3510  3559  3565  3656  4813
.HEADE      1#      2
.KT11       1#    140
.MARHD   2660#  2662
.SCOPE      1#   1833  1876  1915  1955  1989  2050  2062  2093  2114  2135  2163  2353  2410  2455
           2500  2548  2593  2638  2684  2761  2791  2819
.SETUP      1#    188
.SWRHI      1#     13
.SWRLO      1#     25#
```

```
.TM7      2058#   2060
.$ACT1       1#    205
.$APTB       1#    371#
.$APTH       1#    437
.$APTY       1#   4566
.$ASTA       1#    459
.$CATC       1#    188
.$CMTA       1#    323
.$DB2D       1#
.$DB2O       1#
.$DIV        1#
.$EOP        1#   2846
.$ERRO       1#   4028
.$ERRT       1#   4095
.$MULT       1#
.$POWE       1#    281
.$RAND       1#
.$RDDE       1#
.$RDOC       1#   4393
.$READ       1#   4195
.$R2AZ       1#
.$SAVE       1#
.$SB2D       1#
.$SB2O       1#
.$SCOP       1#   3950
.$SIZE       1#
.$SUPR       1#
.$TRAP       1#
.$TYPB       1#
.$TYPD       1#   4659
.$TYPE       1#   4470
.$TYPO       1#   4729
.$40CA       1#
.1170        1#

. ABS.  032000     000

ERRORS DETECTED:  0

CVMSAA,CVMSAA/SOL/CRF=SYSMAC.SML,CVMSAA.P11
RUN-TIME: 21 22 1 SECONDS
RUN-TIME RATIO: 79/44=1.7
CORE USED:  45K  (90 PAGES)
```