Micro Fiche Scan

Name of device(s) tested:
    MSV11-DD/LF/PK/ED/LK/PF/PL/R/QA/QB
Test description:
    0-2 MEG MEM EXER
MAINDEC Number or Package Identifier (after SEP 1977):
    CVMSAB0
Fiche Document Part Number:
    AH-S437B-MC
Fiche preparation date unknown, using copyright year:
    1985
Image resolution:
    1-bit black&white, compressed for minimal file size

IDENTIFICATION

PRODUCT CODE: AC-S435B-MC

PRODUCT NAME: CVMSAB0 0-2 MEG MEM EXER

PRODUCT DATE: DECEMBER 1985

MAINTAINER: ESD DIAGNOSTIC ENGINEERING

REVISION HISTORY

REVISION A:	APRIL 1981	COMMENT
REVISION B:	DECEMBER 1985	11/53 PIPE-LINEING SUPPORT XXDP V2.2 COMPATIBILITY DEPO INCORPORATED UPDATED DOCUMENTATION

TABLE OF CONTENTS

1.0 GENERAL PROGRAM INFORMATION.
1.1 PROGRAM PURPOSE (ABSTRACT)
1.2 SYSTEM REQUIREMENTS
1.2.1 HARDWARE SUPPORTED
1.2.1.1 HARDWARE RESTRICTIONS
1.3 RELATED DOCUMENTS AND STANDARDS
1.4 DIAGNOSTIC HIERARCHY PREREQUISITES
1.5 ASSUMPTIONS

2.0 OPERATING INSTRUCTIONS
2.1 LOADING AND STARTING PROCEDURE
2.2 SPECIAL ENVIRONMENTS
2.3 PROGRAM OPTIONS
2.4 EXECUTION TIMES

3.0 ERROR INFORMATION
3.1 ERROR REPORTING
3.2 ERROR HALTS

4.0 PERFORMANCE AND PROGRESS REPORTS

5.0 DEVICE INFORMATION TABLES

5.2 MOS PARITY REGISTER

6.0 SUB-TEST SUMMARIES
6.1 SECTION 1: ADDRESS TESTS
6.2 SECTION 2: WORST CASE NOISE TESTS
6.3 SECTION 3: INSTRUCTION EXECUTION TESTS
6.4 SECTION 4: MOS TESTS
6.5 SPECIAL TOGGLE IN TESTS

7.0 PROGRAM LISTING

1.0 GENERAL PROGRAM INFORMATION. SEQ 0004

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,LSI-11/53,LSI-11/73 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.

1.2.1 HARDWARE SUPPORTED SEQ 0005

THE FOLLOWING MEMORIES ARE SUPPORTED BY THIS SOFTWARE:
MSV11-DD, MSV11-LF, MSV11-PK, MSV11-ED, MSV11-LK, MSV11 PF,
MSV11 PL, MSV11-R, MSV11-Q/A/B/C, MSV11-S.

1.2.1.1 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. ADDITIONALLY,
THE MSV11-R, MSV11-Q/A/B/C, AND THE MSV11-S ARE SUBJECT
TO THE SAME LIMITATIONS. IT IS RECOMMMENDED THAT THESE
MEMORIES NOT BE MIXED WITH MSV11-D.

1.3 RELATED DOCUMENTS AND STANDARDS

A. PROGRAMMING PRACTICES - DOCUMENT NO. 175-003-009-01
B. PDP-11 MAINDEC SYSMAC PACKAGE - MAINDEC 11-DZQAC-C7-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 PPOGRAM 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 JSER 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 MC. 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.

3.0 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.

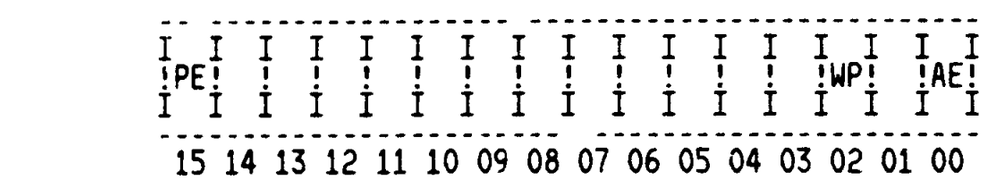
4.0 PERFORMANCE AND PROGRESS REPORTS

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.

J1

6.0 SUB-TEST SUMMARIES SEQ 0009

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  CVMSAB  0-2 MEGAWORD MEMORY EXERCISER, 16K VER
        ;*COPYRIGHT (C) 1985
        ;*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-C7), JUL, 1982.
        ;*
637                             .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>
638                             ;*         7            INHIBIT PROGRAM RELOCATION
                                ;*         6            INHIBIT PARITY ERROR DETECTION
                                ;*         5            INHIBIT EXERCISING VECTOR AREA.
639                             .SBTTL  BASIC DEFINITIONS
                                ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
        001100                  STACK=  1100
        104000                  ERROR=  EMT             ;;BASIC DEFINITION OF ERROR CALL
        000004                  SCOPE=  IOT             ;;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
        177776                  PSW=    PS
        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
```

N1


```
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
001000                  SW9=    SW09
000400                  SW8=    SW08
000200                  SW7=    SW07
000100                  SW6=    SW06
000040                  SW5=    SW05
000020                  SW4=    SW04
000010                  SW3=    SW03
000004                  SW2=    SW02
000002                  SW1=    SW01
000001                  SW0=    SW00
                        ;*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
001000                  BIT9=   BIT09
000400                  BIT8=   BIT08
000200                  BIT7=   BIT07
000100                  BIT6=   BIT06
000040                  BIT5=   BIT05
000020                  BIT4=   BIT04
000010                  BIT3=   BIT03
000004                  BIT2=   BIT02
000002                  BIT1    BIT01
```

```
                        000001                  BIT0=   BIT00
                                                ;*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
    640
    641
    642                                         .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
    643                 000000                  UP = 0                  ;CODE FOR UPWARDS MAP IN MEM MGMT PDR'S
    644                 000006                  RW = 6                  ;CODE FOR READ/WRITE IN MEM MGMT PDR S
    645
    646                                         ;*  PARITY MEMORY DEFINITIONS.
    647                 000001                  AE=1                            ;PARITY ACTION ENABLE
    648                 000114                  PARVEC=114                      ;PARITY TRAP VECTOR
    649
    650                                         ;*  MISCELLANEOUS ASSIGNMENTS
    651                 017777                  MASK4K= 17777                   ;MASK FOR 4K ADDRESS BANK BOUNDARY.
    652                 037777                  MASK8K= 37777                   ;MASK FOR 8K ADDRESS BANK BOUNDARY
    653                 000007                  MFPT=   7                       ;MOVE FROM PROCESSOR TYPE TO R0
    654
    655
    657                                         .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  003744         JMP     @#START ;;JUMP TO STARTING ADDRESS OF PROGRAM
 658 000204      000167  000070         JMP     RESTAR           ;RESTART ADDRESS, USING PREVIOUS PARAMETERS.
 659
 660             000004                 .=ERRVEC
 661 000004      023430                 .WORD   ERRTRP
 662 000006      000000                 .WORD   0
 663
 664                            .SBTTL  ACT11 HOOKS
                                ;;*****************************************************************
                                ;HOOKS REQUIRED BY ACT11
                 000010                 $SVPC=.                  ;SAVE PC
                 000046                 .=46
     000046      012722                 $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     #$CMTAG,SP      ;SET UP THE STACK POINTER.
000312  005767  000216                                      TST     PRGMAP          ;CHECK IF THE MEMORY HAS BEEN MAPPED.
000316  001002                                              BNE     REST2           ;BR IF MEMORY MAPPED.
000320  000167  003434                                      JMP     START           ;GO START,IF NOT MAPPED
000324  005767  000206                              REST2:  TST     MMAVA           ;CHECK IF MEM MGMT AVAILABLE.
000330  001454                                              BEQ     10$             ;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 REG.
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  014546                                      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
000526  000167  006254                                      JMP     START1          ;RESTART WITH PREVIOUSLY SELECTED PARAMETERS.
```

```
                                ;* THE FOLLOWING LOCATIONS ARE USED BY THE ABOVE ROUTINE AND MUST BE LOCATED
                                ;*  BELOW 1000 TO INSURE CORRECT OPERATION UNDER THE WIDEST VARIETY OF
                                ;*  CIRCUMSTANCES.
000532  000000                  RELOCF: .WORD   0       ;CONTAINS RELOCATION FACTOR (NO MEM MGMT)
000534  000000                  PRGMAP: .WORD   0       ;PROGRAM MAP   WHERE THE PROGRAM IS LOCATED
000536  000000                  MMAVA:  .WORD   0       ;MEMORY MANAGEMENT AVAILABLE FLAG.
                                                        ;BIT  0 = 1     MMU AVAILABLE
                                                        ;BIT 15 = 1     22 BIT ADDRESSING AVAILABLE
```

```
733                                     .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  021054                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
000676  024137                  $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.
                                        .=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                  $INTAC: .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)
                                .REPT   4
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     207      377     377 $BELL:  .ASCIZ  <207><377><377> ;;CODE FOR BELL
001167     000
001170     077                  $QUES:  .ASCII  /?/             ;;QUESTION MARK
001171     015                  $CRLF:  .ASCII  <15>            ;;CARRIAGE RETURN
001172     012      000         $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
001202  000000                  $PASS:  .WORD   APASS    ;;PASS COUNT
001204  000000                  $DEVCT: .WORD   ADEVCT   ;;DEVICE COUNT
```

```
001206  000000                  $UNIT:  .WORD   AUNIT   ;;I/O UNIT NUMBER
001210  000000                  $MSGAD: .WORD   AMSGAD  ;;MESSAGE ADDRESS
001212  000000                  $MSGLG: .WORD   AMSGLG  ;;MESSAGE LENGTH
001214                          $ETABLE:                ;;APT ENVIRONMENT TABLE
001214     000                  $ENV:   .BYTE   AENV    ;;ENVIRONMENT BYTE
001215     000                  $ENVM:  .BYTE   AENVM   ;;ENVIRONMENT MODE BITS
001216  000000                  $SWREG: .WORD   ASWREG  ;;APT SWITCH REGISTER
001220  000000                  $USWR:  .WORD   AUSWR   ;;USER SWITCHES
001222  000000                  $CPUOP: .WORD   ACPUOP  ;;CPU TYPE,OPTIONS
                                ;*                      BITS 15-11=CPU TYPE
                                ;*                             11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
                                ;*                             11/70=06,PDQ=07,Q=10
                                ;*                      BIT 10=REAL TIME CLOCK
                                ;*                      BIT  9=FLOATING POINT PROCESSOR
                                ;*                      BIT  8=MEMORY MANAGEMENT
001224     000                  $MAMS1: .BYTE   AMAMS1  ;;HIGH ADDRESS,M.S. BYTE
001225     000                  $MTYP1: .BYTE   AMTYP1  ;;MEM. TYPE,BLK#1
                                ;*                      MEM.TYPE BYTE   --  (HIGH BYTE)
                                ;*                             900 NSEC CORE=001
                                ;*                             300 NSEC BIPOLAR=002
                                ;*                             500 NSEC MOS=003
001226  000000                  $MADR1: .WORD   AMADR1  ;;HIGH ADDRESS,BLK#1
                                ;*                       MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
001230     000                  $MAMS2: .BYTE   AMAMS2  ;;HIGH ADDRESS,M.S. BYTE
001231     000                  $MTYP2: .BYTE   AMTYP2  ;;MEM.TYPE,BLK#2
001232  000000                  $MADR2: .WORD   AMADR2  ;;MEM.LAST ADDRESS,BLK#2
001234     000                  $MAMS3: .BYTE   AMAMS3  ;;HIGH ADDRESS,M.S.BYTE
001235     000                  $MTYP3: .BYTE   AMTYP3  ;;MEM.TYPE,BLK#3
001236  000000                  $MADR3: .WORD   AMADR3  ;;MEM.LAST ADDRESS,BLK#3
001240     000                  $MAMS4: .BYTE   AMAMS4  ;;HIGH ADDRESS,M.S.BYTE
001241     000                  $MTYP4: .BYTE   AMTYP4  ;;MEM.TYPE,BLK#4
001242  000000                  $MADR4: .WORD   AMADR4  ;;MEM.LAST ADDRESS,BLK#4
001244  000000                  $VECT1: .WORD   AVECT1  ;;INTERRUPT VECTOR#1,BUS PRIORITY#1
001246  000000                  $VECT2: .WORD   AVECT2  ;;INTERRUPT VECTOR#2BUS PRIORITY#2
001250  000000                  $BASE:  .WORD   ABASE   ;;BASE ADDRESS OF EQUIPMENT UNDER TEST
001252  000000                  $DEVM:  .WORD   ADEVM   ;;DEVICE MAP
001254  000000                  $CDW1:  .WORD   ACDW1   ;;CONTROLLER DESCRIPTION WORD#1
001256  000000                  $CDW2:  .WORD   ACDW2   ;;CONTROLLER DESCRIPTION WORD#2
001260  000000                  $DDW0:  .WORD   ADDW0   ;;DEVICE DESCRIPTOR WORD#0
001262  000000                  $DDW1:  .WORD   ADDW1   ;;DEVICE DESCRIPTOR WORD#1
001264  000000                  $DDW2:  .WORD   ADDW2   ;;DEVICE DESCRIPTOR WORD#2
001266  000000                  $DDW3:  .WORD   ADDW3   ;;DEVICE DESCRIPTOR WORD#3
001270  000000                  $DDW4:  .WORD   ADDW4   ;;DEVICE DESCRIPTOR WORD#4
001272  000000                  $DDW5:  .WORD   ADDW5   ;;DEVICE DESCRIPTOR WORD#5
001274  000000                  $DDW6:  .WORD   ADDW6   ;;DEVICE DESCRIPTOR WORD#6
001276  000000                  $DDW7:  .WORD   ADDW7   ;;DEVICE DESCRIPTOR WORD#7
001300  000000                  $DDW8:  .WORD   ADDW8   ;;DEVICE DESCRIPTOR WORD#8
001302  000000                  $DDW9:  .WORD   ADDW9   ;;DEVICE DESCRIPTOR WORD#9
001304  000000                  $DDW10: .WORD   ADDW10  ;;DEVICE DESCRIPTOR WORD#10
001306  000000                  $DDW11: .WORD   ADDW11  ;;DEVICE DESCRIPTOR WORD#11
001310  000000                  $DDW12: .WORD   ADDW12  ;;DEVICE DESCRIPTOR WORD#12
001312  000000                  $DDW13: .WORD   ADDW13  ;;DEVICE DESCRIPTOR WORD#13
001314  000000                  $DDW14: .WORD   ADDW14  ;;DEVICE DESCRIPTOR WORD#14
001316  000000                  $DDW15: .WORD   ADDW15  ;;DEVICE DESCRIPTOR WORD#15
001320                          $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:
000031                                  .REPT   31
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
001630  000000                          .WORD   0       ;7TH  WORD MAP...896KW
001632  000000                          .WORD   0       ;8TH  WORD MAP..1024KW
001634  000000                          .WORD   0       ;9TH  WORD MAP..1152KW
001636  000000                          .WORD   0       ;10TH WORD MAP..1280KW
001640  000000                          .WORD   0       ;11TH WORD MAP..1408KW
001642  000000                          .WORD   0       ;12TH WORD MAP..1536KW
001644  000000                          .WORD   0       ;13TH WORD MAP..1664KW
```

```
001646  000000                          .WORD   0       ;14TH WORD MAP..1792KW
001650  000000                          .WORD   0       ;15TH WORD MAP..1920KW
001652  000000                          .WORD   0       ;16TH WORD MAP..2048KW

001654                          PMEMAP:                 ;PARITY MAP - WHICH BANKS HAVE MEMORY PARITY

001654  000000                          .WORD   0       ;1ST  WORD MAP...128KW
001656  000000                          .WORD   0       ;2ND  WORD MAP...256KW
001660  000000                          .WORD   0       ;3RD  WORD MAP...384KW
001662  000000                          .WORD   0       ;4TH  WORD MAP...512KW
001664  000000                          .WORD   0       ;5TH  WORD MAP...640KW
001666  000000                          .WORD   0       ;6TH  WORD MAP...768KW
001670  000000                          .WORD   0       ;7TH  WORD MAP...896KW
001672  000000                          .WORD   0       ;8TH  WORD MAP..1024KW
001674  000000                          .WORD   0       ;9TH  WORD MAP..1152KW
001676  000000                          .WORD   0       ;10TH WORD MAP..1280KW
001700  000000                          .WORD   0       ;11TH WORD MAP..1408KW
001702  000000                          .WORD   0       ;12TH WORD MAP..1536KW
001704  000000                          .WORD   0       ;13TH WORD MAP..1664KW
001706  000000                          .WORD   0       ;14TH WORD MAP..1792KW
001710  000000                          .WORD   0       ;15TH WORD MAP..1920KW
001712  000000                          .WORD   0       ;16TH WORD MAP..2048KW

001714                          BITPT:                  ;POINTER TO CURRENT 8K BANK OF MEMORY

001714  000000                          .WORD   0       ;1ST  WORD MAP...128KW
001716  000000                          .WORD   0       ;2ND  WORD MAP...256KW
001720  000000                          .WORD   0       ;3RD  WORD MAP...384KW
001722  000000                          .WORD   0       ;4TH  WORD MAP...512KW
001724  000000                          .WORD   0       ;5TH  WORD MAP...640KW
001726  000000                          .WORD   0       ;6TH  WORD MAP...768KW
001730  000000                          .WORD   0       ;7TH  WORD MAP...896KW
001732  000000                          .WORD   0       ;8TH  WORD MAP..1024KW
001734  000000                          .WORD   0       ;9TH  WORD MAP..1152KW
001736  000000                          .WORD   0       ;10TH WORD MAP..1280KW
001740  000000                          .WORD   0       ;11TH WORD MAP..1408KW
001742  000000                          .WORD   0       ;12TH WORD MAP..1536KW
001744  000000                          .WORD   0       ;13TH WORD MAP..1664KW
001746  000000                          .WORD   0       ;14TH WORD MAP..1792KW
001750  000000                          .WORD   0       ;15TH WORD MAP..1920KW
001752  000000                          .WORD   0       ;16TH WORD MAP..2048KW



001754  000000                  MMORE:  .WORD   0       ;LOOP ADDRESS FOR MULTIPLE BLOCK TESTING.
                                                        ;SET UP BY "INITMM" AND "INITDN" ROUTINES.
                                                        ;USED BY "MMUP" AND "MMDOWN" ROUTINES.
001756     000                  SELFLG: .BYTE   0       ;OPERATOR SELECTED PARAMETERS FLAG.  (SA=204)
001757     000                  FLAG8K: .BYTE   0       ;8K BLOCK INDICATOR.  USED IN "INITMM" AND "MMUP .
001760     000                  OEFLG:  .BYTE   0       ;ODD/EVEN FLAG USED IN PARITY MEMORY BYTE TEST.
                                        .EVEN
001762  000000                  FSTADR: .WORD   0       ;FIRST VIRTUAL ADDRESS TO BE TESTED.
                                                        ;FIRST ADDRESS IS USER SELECTABLE.
                                                        ; TO BREAK TO 'MMUP" TO FIND LAST ADDRESS.
001764  000000                  .CONST: .WORD   0       ;USER SELECTABLE CONSTANT DATA.
001766  000004                  WWP:    .WORD   4       ;WRITE WRONG PARITY COMMAND
001770  000000                  TEMP:   .WORD   0       ;TEMPORARY STORAGE
```

L2


```
001772  000000                  TEMP1:  .WORD   0       ;TEMPORARY STORAGE
001774  000000                  FLG30K: .WORD   0       ;30K MEMORY FLAG
001776  000000                  LSIFLG: .WORD   0       ;LSI-11 PROCESSOR FLAG
002000  000000                  KDJDA:  .WORD   0       ;;VK KDJ11-DA (11/53) CPU FLAG
002002  000000                  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.
                                ;;*****************************************************************
002004                          RADTAB:
002004  001070                  .STACK: $CMTAG          ;STACK POINTER INITIAL ADDRESS.
002006  001514                  .MEMMAP:        MEMMAP  ;AUTO. MEMORY SIZING MAP
002010  001614                  .SAVTST:        SAVTST  ;SIZED MEMORY MAP FROM AUTO. OR USER
002012  001554                  .TSTMAP:        TSTMAP  ;MEMORY MAP OF 8K BANKS TO TEST
002014  001654                  .PMEMAP:        PMEMAP  ;PARITY MAP OF 8K BANKS
002016  001714                  .BITPT: BITPT           ;BIT POINTER FOR BLOCK UNDER TEST
002020  001506                  .RESRV: RESRVD          ;PARITY REGISTER RESERVED BIT MASK ADDRESS.
002022  002272                  .MPR0:  MPR0            ;MEMORY PARITY REGISTER TABLE ADDRESS.
002024  003372                  .MPRX:  MPRX            ;MEMORY PARITY REGISTER EXIST TABLE ADDRESS.
002026  010520                  .PBTRP: PBTRP           ;PARITY BYTE TEST TRAP ROUTINE ADDRESS
002030  002244                  .MPPAT: MPPATS          ;MEMORY PARITY PATTERN TABLE ADDRESS.
002032  015632                  .PESRV: PESRV           ;MEMORY PARITY ERROR TRAP ROUTINE ADDRESS.
002034  003434                  .ERRTB: $ERRTB          ;ERROR TYPEOUT TABLE PONTER.
002036  000010                  .EIGHT: 8.              ;DECIMAL TYPE ROUTINE COUNT DESIGNATOR.
002040  012506                  .TST32: TST32           ;SCOPE ABORT ADR FOR WHEN NO MEM AVA FOR TEST.
                                ;;*****************************************************************
                                ;*  DATA CONTAINERS FOR ERROR PRINTOUT.
                                ;;*****************************************************************
002042  001106  001110  001114  DT1:    $ERRPC,$GDADR,$GDDAT,$BDDAT,0
002050  001116  000000
002054  001504  001106  001110  DT2:    $VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT,0
002062  001114  001116  000000
002070  001504  001106  001110  DT12:   $VERPC,$ERRPC,$GDADR,$GDDAT,0
002076  001114  000000
002102  001504  001106  001150  DT14:   $VERPC,$ERRPC,$TMP0,$GDADR,0
002110  001110  000000
002114  001504  001106  001110  DT15:   $VERPC,$ERRPC,$GDADR,$TMP0,$GDDAT,$BDDAT,0
002122  001150  001114  001116
002130  000000
002132  001504  001106  001150  DT21:   $VERPC,$ERRPC,$TMP0,$GDADR,$GDDAT,$BDDAT,0
002140  001110  001114  001116
002146  000000
002150  001504  001106  001110  DT23:   $VERPC,$ERRPC,$GDADR,$BDADR,$GDDAT,$BDDAT,0
002156  001112  001114  001116
002164  000000
002166  001504  001106  001112  DT24:   $VERPC,$ERRPC,$BDADR,0
002174  000000
002176  001504  001106  001112  DT25:   $VERPC,$ERRPC,$BDADR,$TMP0,$TMP1,0
002204  001150  001152  000000
002212  001504  001106  001150  DT26:   $VERPC,$ERRPC,$TMP0,$TMP1,0
002220  001152  000000
002224  001150  001152  001110  DT30:   $TMP0,$TMP1,$GDADR,$BDDAT,0
002232  001116  000000
002236  001156  000000          DT31:   $TMP3,0
```

```
002242  177777                          .WORD    1      ;TABLE TERMINATOR.

                                .SBTTL  MEMORY PARITY PATTERNS TABLE
                                ;;*****************************************************************
                                ;THE FOLLOWING ARE THE PARITY PATTERNS EXERCISED THRUOUT MEMORY
                                ;;*****************************************************************

002244  125325                  MPPATS: 125325                  ;EVEN,ODD
002246  152652                          152652                  ;ODD,EVEN
002250  052452                          052452                  ;EVEN,ODD
002252  025125                          025125                  ;ODD,EVEN
002254  102070                          102070                  ;EVEN,EVEN
002256  072527                          072527                  ;ODD,ODD
002260  177777                          177777                  ;EVEN,EVEN
002262  107030                          107030                  ;ODD,ODD
002264  152525                          152525                  ;ODD,EVEN
002266  000000                          0                       ;EXTRA PATTERN HOLDER FOR
                                                                ;FUTURE USE
002270  000000                  MPEND:  0                       ;TABLE TERMINATOR

                                .SBTTL  MEMORY PARITY REGISTER ADDRESS TABLE
                                ;////////////////////////////////////////////////////////////////
                                ;*  THE FOLLOWING REPRESENTS THE MEMORY PARITY REGISTER ADDRESS TABLE
                                ;*  FROM WHICH PARITY MEMORY IS ADDRESSED & CONTROLLED:
                                ;*
                                ;*  THE LEAST SIGNIFICANT BIT IN THE DEVICE ADDRESS IS SET TO A ONE (1)
                                ;*  IF THE CONTROL IS FOUND NOT TO BE PRESENT.  THE MEMORY PRESENT UNDER
                                ;*  THE CONTROL OF EACH CONTROLLER IS REPRESENTED BY 16 WORDS FOLLOWING
                                ;*  THE DEVICE ADDRESS.  FOR NON-22BIT MEMORY ONLY 1 WORD IS USED
                                ;*  (EACH BIT REPRESENTING A 8K BLOCK,  I.E.  FIRST WORD BIT0 = 0 - 8K,
                                ;*      BIT  3 =        24-28KW         16 BIT ADDRESSING
                                ;*      BIT 15 =        120-124KW       18 BIT ADDRESSING
                                ;////////////////////////////////////////////////////////////////

002272  172101                  MPR0:   172100 +1               ;PARITY STATUS REGISTER
002274  000000                          .WORD   0               ;1ST  WORD CONTROL MAP...128KW
002276  000000                          .WORD   0               ;2ND  WORD CONTROL MAP...256KW
002300  000000                          .WORD   0               ;3RD  WORD CONTROL MAP...384KW
002302  000000                          .WORD   0               ;4TH  WORD CONTROL MAP...512KW
002304  000000                          .WORD   0               ;5TH  WORD CONTROL MAP...640KW
002306  000000                          .WORD   0               ;6TH  WORD CONTROL MAP...768KW
002310  000000                          .WORD   0               ;7TH  WORD CONTROL MAP...896KW
002312  000000                          .WORD   0               ;8TH  WORD CONTROL MAP..1024KW
002314  000000                          .WORD   0               ;9TH  WORD CONTROL MAP..1152KW
002316  000000                          .WORD   0               ;10TH WORD CONTROL MAP..1280KW
002320  000000                          .WORD   0               ;11TH WORD CONTROL MAP..1408KW
002322  000000                          .WORD   0               ;12TH WORD CONTROL MAP..1536KW
002324  000000                          .WORD   0               ;13TH WORD CONTROL MAP..1664KW
002326  000000                          .WORD   0               ;14TH WORD CONTROL MAP..1792KW
002330  000000                          .WORD   0               ;15TH WORD CONTROL MAP..1920KW
002332  000000                          .WORD   0               ;16TH WORD CONTROL MAP..2048KW
002334  000000                          .WORD   0               ;MASK FOR UNUSED PARITY CSR BITS
002336  172103                  MPR1:   172102 +1               ;PARITY STATUS REGISTER
002340  000000                          .WORD   0               ;1ST  WORD CONTROL MAP...128KW
002342  000000                          .WORD   0               ;2ND  WORD CONTROL MAP...256KW
002344  000000                          .WORD   0               ;3RD  WORD CONTROL MAP...384KW
002346  000000                          .WORD   0               ;4TH  WORD CONTROL MAP...512KW
002350  000000                          .WORD   0               ;5TH  WORD CONTROL MAP...640KW
```

N2


```
002352  000000                          .WORD   0       ;6TH  WORD CONTROL MAP...768KW
002354  000000                          .WORD   0       ;7TH  WORD CONTROL MAP...896KW
002356  000000                          .WORD   0       ;8TH  WORD CONTROL MAP..1024KW
002360  000000                          .WORD   0       ;9TH  WORD CONTROL MAP..1152KW
002362  000000                          .WORD   0       ;10TH WORD CONTROL MAP..1280KW
002364  000000                          .WORD   0       ;11TH WORD CONTROL MAP..1408KW
002366  000000                          .WORD   0       ;12TH WORD CONTROL MAP..1536KW
002370  000000                          .WORD   0       ;13TH WORD CONTROL MAP..1664KW
002372  000000                          .WORD   0       ;14TH WORD CONTROL MAP..1792KW
002374  000000                          .WORD   0       ;15TH WORD CONTROL MAP..1920KW
002376  000000                          .WORD   0       ;16TH WORD CONTROL MAP..2048KW
002400  000000                          .WORD   0       ;MASK FOR UNUSED PARITY CSR BITS
002402  172105                  MPR2:   172104 +1       ;PARITY STATUS REGISTER
002404  000000                          .WORD   0       ;1ST  WORD CONTROL MAP...128KW
002406  000000                          .WORD   0       ;2ND  WORD CONTROL MAP...256KW
002410  000000                          .WORD   0       ;3RD  WORD CONTROL MAP...384KW
002412  000000                          .WORD   0       ;4TH  WORD CONTROL MAP...512KW
002414  000000                          .WORD   0       ;5TH  WORD CONTROL MAP...640KW
002416  000000                          .WORD   0       ;6TH  WORD CONTROL MAP...768KW
002420  000000                          .WORD   0       ;7TH  WORD CONTROL MAP...896KW
002422  000000                          .WORD   0       ;8TH  WORD CONTROL MAP..1024KW
002424  000000                          .WORD   0       ;9TH  WORD CONTROL MAP..1152KW
002426  000000                          .WORD   0       ;10TH WORD CONTROL MAP..1280KW
002430  000000                          .WORD   0       ;11TH WORD CONTROL MAP..1408KW
002432  000000                          .WORD   0       ;12TH WORD CONTROL MAP..1536KW
002434  000000                          .WORD   0       ;13TH WORD CONTROL MAP..1664KW
002436  000000                          .WORD   0       ;14TH WORD CONTROL MAP..1792KW
002440  000000                          .WORD   0       ;15TH WORD CONTROL MAP..1920KW
002442  000000                          .WORD   0       ;16TH WORD CONTROL MAP..2048KW
002444  000000                          .WORD   0       ;MASK FOR UNUSED PARITY CSR BITS
002446  172107                  MPR3:   172106 +1       ;PARITY STATUS REGISTER
002450  000000                          .WORD   0       ;1ST  WORD CONTROL MAP.. 128KW
002452  000000                          .WORD   0       ;2ND  WORD CONTROL MAP...256KW
002454  000000                          .WORD   0       ;3RD  WORD CONTROL MAP...384KW
002456  000000                          .WORD   0       ;4TH  WORD CONTROL MAP...512KW
002460  000000                          .WORD   0       ;5TH  WORD CONTROL MAP...640KW
002462  000000                          .WORD   0       ;6TH  WORD CONTROL MAP...768KW
002464  000000                          .WORD   0       ;7TH  WORD CONTROL MAP...896KW
002466  000000                          .WORD   0       ;8TH  WORD CONTROL MAP..1024KW
002470  000000                          .WORD   0       ;9TH  WORD CONTROL MAP..1152KW
002472  000000                          .WORD   0       ;10TH WORD CONTROL MAP..1280KW
002474  000000                          .WORD   0       ;11TH WORD CONTROL MAP..1408KW
002476  000000                          .WORD   0       ;12TH WORD CONTROL MAP..1536KW
002500  000000                          .WORD   0       ;13TH WORD CONTROL MAP..1664KW
002502  000000                          .WORD   0       ;14TH WORD CONTROL MAP..1792KW
002504  000000                          .WORD   0       ;15TH WORD CONTROL MAP..1920KW
002506  000000                          .WORD   0       ;16TH WORD CONTROL MAP..2048KW
002510  000000                          .WORD   0       ;MASK FOR UNUSED PARITY CSR BITS
002512  172111                  MPR4:   172110 +1       ;PARITY STAUS REGISTER
002514  000000                          .WORD   0       ;1ST  WORD CONTROL MAP...128KW
002516  000000                          .WORD   0       ;2ND  WORD CONTROL MAP...256KW
002520  000000                          .WORD   0       ;3RD  WORD CONTROL MAP...384KW
002522  000000                          .WORD   0       ;4TH  WORD CONTROL MAP...512KW
002524  000000                          .WORD   0       ;5TH  WORD CONTROL MAP...640KW
002526  000000                          .WORD   0       ;6TH  WORD CONTROL MAP...768KW
002530  000000                          .WORD   0       ;7TH  WORD CONTROL MAP...896KW
002532  000000                          .WORD   0       ;8TH  WORD CONTROL MAP..1024KW
```

```
002534  000000                          .WORD   0       ;9TH  WORD CONTROL MAP..1152KW
002536  000000                          .WORD   0       ;10TH WORD CONTROL MAP..1280KW
002540  000000                          .WORD   0       ;11TH WORD CONTROL MAP..1408KW
002542  000000                          .WORD   0       ;12TH WORD CONTROL MAP..1536KW
002544  000000                          .WORD   0       ;13TH WORD CONTROL MAP..1664KW
002546  000000                          .WORD   0       ;14TH WORD CONTROL MAP..1792KW
002550  000000                          .WORD   0       ;15TH WORD CONTROL MAP..1920KW
002552  000000                          .WORD   0       ;16TH WORD CONTROL MAP..2048KW
002554  000000                          .WORD   0       ;MASK FOR UNUSED PARITY CSR BITS
002556  172113                  MPR5:   172112 +1       ;PARITY STATUS REGISTER
002560  000000                          .WORD   0       ;1ST  WORD CONTROL MAP...128KW
002562  000000                          .WORD   0       ;2ND  WORD CONTROL MAP...256KW
002564  000000                          .WORD   0       ;3RD  WORD CONTROL MAP...384KW
002566  000000                          .WORD   0       ;4TH  WORD CONTROL MAP...512KW
002570  000000                          .WORD   0       ;5TH  WORD CONTROL MAP...640KW
002572  000000                          .WORD   0       ;6TH  WORD CONTROL MAP...768KW
002574  000000                          .WORD   0       ;7TH  WORD CONTROL MAP...896KW
002576  000000                          .WORD   0       ;8TH  WORD CONTROL MAP..1024KW
002600  000000                          .WORD   0       ;9TH  WORD CONTROL MAP..1152KW
002602  000000                          .WORD   0       ;10TH WORD CONTROL MAP..1280KW
002604  000000                          .WORD   0       ;11TH WORD CONTROL MAP..1408KW
002606  000000                          .WORD   0       ;12TH WORD CONTROL MAP..1536KW
002610  000000                          .WORD   0       ;13TH WORD CONTROL MAP..1664KW
002612  000000                          .WORD   0       ;14TH WORD CONTROL MAP..1792KW
002614  000000                          .WORD   0       ;15TH WORD CONTROL MAP..1920KW
002616  000000                          .WORD   0       ;16TH WORD CONTROL MAP..2048KW
002620  000000                          .WORD   0       ;MASK FOR UNUSED PARITY CSR BITS
002622  172115                  MPR6:   172114 +1       ;PARITY STATUS REGISTER
002624  000000                          .WORD   0       ;1ST  WORD CONTROL MAP...128KW
002626  000000                          .WORD   0       ;2ND  WORD CONTROL MAP...256KW
002630  000000                          .WORD   0       ;3RD  WORD CONTROL MAP...384KW
002632  000000                          .WORD   0       ;4TH  WORD CONTROL MAP...512KW
002634  000000                          .WORD   0       ;5TH  WORD CONTROL MAP...640KW
002636  000000                          .WORD   0       ;6TH  WORD CONTROL MAP...768KW
002640  000000                          .WORD   0       ;7TH  WORD CONTROL MAP...896KW
002642  000000                          .WORD   0       ;8TH  WORD CONTROL MAP..1024KW
002644  000000                          .WORD   0       ;9TH  WORD CONTROL MAP..1152KW
002646  000000                          .WORD   0       ;10TH WORD CONTROL MAP..1280KW
002650  000000                          .WORD   0       ;11TH WORD CONTROL MAP..1408KW
002652  000000                          .WORD   0       ;12TH WORD CONTROL MAP..1536KW
002654  000000                          .WORD   0       ;13TH WORD CONTROL MAP..1664KW
002656  000000                          .WORD   0       ;14TH WORD CONTROL MAP..1792KW
002660  000000                          .WORD   0       ;15TH WORD CONTROL MAP..1920KW
002662  000000                          .WORD   0       ;16TH WORD CONTROL MAP..2048KW
002664  000000                          .WORD   0       ;MASK FOR UNUSED PARITY CSR BITS
002666  172117                  MPR7:   172116 +1       ;PARITY STATUS REGISTER
002670  000000                          .WORD   0       ;1ST  WORD CONTROL MAP...128KW
002672  000000                          .WORD   0       ;2ND  WORD CONTROL MAP...256KW
002674  000000                          .WORD   0       ;3RD  WORD CONTROL MAP...384KW
002676  000000                          .WORD   0       ;4TH  WORD CONTROL MAP...512KW
002700  000000                          .WORD   0       ;5TH  WORD CONTROL MAP...640KW
002702  000000                          .WORD   0       ;6TH  WORD CONTROL MAP...768KW
002704  000000                          .WORD   0       ;7TH  WORD CONTROL MAP...896KW
002706  000000                          .WORD   0       ;8TH  WORD CONTROL MAP..1024KW
002710  000000                          .WORD   0       ;9TH  WORD CONTROL MAP..1152KW
002712  000000                          .WORD   0       ;10TH WORD CONTROL MAP..1280KW
002714  000000                          .WORD   0       ;11TH WORD CONTROL MAP..1408KW
```

```
002716  000000                          .WORD   0       ;12TH WORD CONTROL MAP..1536KW
002720  000000                          .WORD   0       ;13TH WORD CONTROL MAP..1664KW
002722  000000                          .WORD   0       ;14TH WORD CONTROL MAP..1792KW
002724  000000                          .WORD   0       ;15TH WORD CONTROL MAP..1920KW
002726  000000                          .WORD   0       ;16TH WORD CONTROL MAP..2048KW
002730  000000                          .WORD   0       ;MASK FOR UNUSED PARITY CSR BITS
002732  172121                  MPR8:   172120 +1       ;PARITY STATUS REGISTER
002734  000000                          .WORD   0       ;1ST  WORD CONTROL MAP...128KW
002736  000000                          .WORD   0       ;2ND  WORD CONTROL MAP...256KW
002740  000000                          .WORD   0       ;3RD  WORD CONTROL MAP...384KW
002742  000000                          .WORD   0       ;4TH  WORD CONTROL MAP...512KW
002744  000000                          .WORD   0       ;5TH  WORD CONTROL MAP...640KW
002746  000000                          .WORD   0       ;6TH  WORD CONTROL MAP...768KW
002750  000000                          .WORD   0       ;7TH  WORD CONTROL MAP...896KW
002752  000000                          .WORD   0       ;8TH  WORD CONTROL MAP..1024KW
002754  000000                          .WORD   0       ;9TH  WORD CONTROL MAP..1152KW
002756  000000                          .WORD   0       ;10TH WORD CONTROL MAP..1280KW
002760  000000                          .WORD   0       ;11TH WORD CONTROL MAP..1408KW
002762  000000                          .WORD   0       ;12TH WORD CONTROL MAP..1536KW
002764  000000                          .WORD   0       ;13TH WORD CONTROL MAP..1664KW
002766  000000                          .WORD   0       ;14TH WORD CONTROL MAP..1792KW
002770  000000                          .WORD   0       ;15TH WORD CONTROL MAP..1920KW
002772  000000                          .WORD   0       ;16TH WORD CONTROL MAP..2048KW
002774  000000                          .WORD   0       ;MASK FOR UNUSED PARITY CSR BITS
002776  172123                  MPR9:   172122 +1       ;PARITY STATUS REGISTER
003000  000000                          .WORD   0       ;1ST  WORD CONTROL MAP...128KW
003002  000000                          .WORD   0       ;2ND  WORD CONTROL MAP...256KW
003004  000000                          .WORD   0       ;3RD  WORD CONTROL MAP...384KW
003006  000000                          .WORD   0       ;4TH  WORD CONTROL MAP...512KW
003010  000000                          .WORD   0       ;5TH  WORD CONTROL MAP...640KW
003012  000000                          .WORD   0       ;6TH  WORD CONTROL MAP...768KW
003014  000000                          .WORD   0       ;7TH  WORD CONTROL MAP...896KW
003016  000000                          .WORD   0       ;8TH  WORD CONTROL MAP..1024KW
003020  000000                          .WORD   0       ;9TH  WORD CONTROL MAP..1152KW
003022  000000                          .WORD   0       ;10TH WORD CONTROL MAP..1280KW
003024  000000                          .WORD   0       ;11TH WORD CONTROL MAP..1408KW
003026  000000                          .WORD   0       ;12TH WORD CONTROL MAP..1536KW
003030  000000                          .WORD   0       ;13TH WORD CONTROL MAP..1664KW
003032  000000                          .WORD   0       ;14TH WORD CONTROL MAP..1792KW
003034  000000                          .WORD   0       ;15TH WORD CONTROL MAP..1920KW
003036  000000                          .WORD   0       ;16TH WORD CONTROL MAP..2048KW
003040  000000                          .WORD   0       ;MASK FOR UNUSED PARITY CSR BITS
003042  172125                  MPR10:  172124 +1       ;PARITY STATUS REGISTER
003044  000000                          .WORD   0       ;1ST  WORD CONTROL MAP...128KW
003046  000000                          .WORD   0       ;2ND  WORD CONTROL MAP...256KW
003050  000000                          .WORD   0       ;3RD  WORD CONTROL MAP...384KW
003052  000000                          .WORD   0       ;4TH  WORD CONTROL MAP...512KW
003054  000000                          .WORD   0       ;5TH  WORD CONTROL MAP...640KW
003056  000000                          .WORD   0       ;6TH  WORD CONTROL MAP...768KW
003060  000000                          .WORD   0       ;7TH  WORD CONTROL MAP...896KW
003062  000000                          .WORD   0       ;8TH  WORD CONTROL MAP..1024KW
003064  000000                          .WORD   0       ;9TH  WORD CONTROL MAP..1152KW
003066  000000                          .WORD   0       ;10TH WORD CONTROL MAP..1280KW
003070  000000                          .WORD   0       ;11TH WORD CONTROL MAP..1408KW
003072  000000                          .WORD   0       ;12TH WORD CONTROL MAP..1536KW
003074  000000                          .WORD   0       ;13TH WORD CONTROL MAP..1664KW
003076  000000                          .WORD   0       ;14TH WORD CONTROL MAP..1792KW
```

```
003100  000000                      .WORD   0       ;15TH WORD CONTROL MAP..1920KW
003102  000000                      .WORD   0       ;16TH WORD CONTROL MAP..2048KW
003104  000000                      .WORD   0       ;MASK FOR UNUSED PARITY CSR BITS
003106  172127              MPR11:  172126 +1       ;PARITY STATUS REGISTER
003110  000000                      .WORD   0       ;1ST  WORD CONTROL MAP...128KW
003112  000000                      .WORD   0       ;2ND  WORD CONTROL MAP...256KW
003114  000000                      .WORD   0       ;3RD  WORD CONTROL MAP...384KW
003116  000000                      .WORD   0       ;4TH  WORD CONTROL MAP...512KW
003120  000000                      .WORD   0       ;5TH  WORD CONTROL MAP...640KW
003122  000000                      .WORD   0       ;6TH  WORD CONTROL MAP...768KW
003124  000000                      .WORD   0       ;7TH  WORD CONTROL MAP...896KW
003126  000000                      .WORD   0       ;8TH  WORD CONTROL MAP..1024KW
003130  000000                      .WORD   0       ;9TH  WORD CONTROL MAP..1152KW
003132  000000                      .WORD   0       ;10TH WORD CONTROL MAP..1280KW
003134  000000                      .WORD   0       ;11TH WORD CONTROL MAP..1408KW
003136  000000                      .WORD   0       ;12TH WORD CONTROL MAP..1536KW
003140  000000                      .WORD   0       ;13TH WORD CONTROL MAP..1664KW
003142  000000                      .WORD   0       ;14TH WORD CONTROL MAP..1792KW
003144  000000                      .WORD   0       ;15TH WORD CONTROL MAP..1920KW
003146  000000                      .WORD   0       ;16TH WORD CONTROL MAP..2048KW
003150  000000                      .WORD   0       ;MASK FOR UNUSED PARITY CSR BITS
003152  172131              MPR12:  172130 +1       ;PARITY STATUS REGISTER
003154  000000                      .WORD   0       ;1ST  WORD CONTROL MAP...128KW
003156  000000                      .WORD   0       ;2ND  WORD CONTROL MAP...256KW
003160  000000                      .WORD   0       ;3RD  WORD CONTROL MAP...384KW
003162  000000                      .WORD   0       ;4TH  WORD CONTROL MAP...512KW
003164  000000                      .WORD   0       ;5TH  WORD CONTROL MAP...640KW
003166  000000                      .WORD   0       ;6TH  WORD CONTROL MAP...768KW
003170  000000                      .WORD   0       ;7TH  WORD CONTROL MAP...896KW
003172  000000                      .WORD   0       ;8TH  WORD CONTROL MAP..1024KW
003174  000000                      .WORD   0       ;9TH  WORD CONTROL MAP..1152KW
003176  000000                      .WORD   0       ;10TH WORD CONTROL MAP..1280KW
003200  000000                      .WORD   0       ;11TH WORD CONTROL MAP..1408KW
003202  000000                      .WORD   0       ;12TH WORD CONTROL MAP..1536KW
003204  000000                      .WORD   0       ;13TH WORD CONTROL MAP..1664KW
003206  000000                      .WORD   0       ;14TH WORD CONTROL MAP..1792KW
003210  000000                      .WORD   0       ;15TH WORD CONTROL MAP..1920KW
003212  000000                      .WORD   0       ;16TH WORD CONTROL MAP..2048KW
003214  000000                      .WORD   0       ;MASK FOR UNUSED PARITY CSR BITS
003216  172133              MPR13:  172132 +1       ;PARITY STATUS REGISTER
003220  000000                      .WORD   0       ;1ST  WORD CONTROL MAP...128KW
003222  000000                      .WORD   0       ;2ND  WORD CONTROL MAP...256KW
003224  000000                      .WORD   0       ;3RD  WORD CONTROL MAP...384KW
003226  000000                      .WORD   0       ;4TH  WORD CONTROL MAP...512KW
003230  000000                      .WORD   0       ;5TH  WORD CONTROL MAP ..640KW
003232  000000                      .WORD   0       ;6TH  WORD CONTROL MAP...768KW
003234  000000                      .WORD   0       ;7TH  WORD CONTROL MAP...896KW
003236  000000                      .WORD   0       ;8TH  WORD CONTROL MAP..1024KW
003240  000000                      .WORD   0       ;9TH  WORD CONTROL MAP..1152KW
003242  000000                      .WORD   0       ;10TH WORD CONTROL MAP..1280KW
003244  000000                      .WORD   0       ;11TH WORD CONTROL MAP..1408KW
003246  000000                      .WORD   0       ;12TH WORD CONTROL MAP..1536KW
003250  000000                      .WORD   0       ;13TH WORD CONTROL MAP..1664KW
003252  000000                      .WORD   0       ;14TH WORD CONTROL MAP..1792KW
003254  000000                      .WORD   0       ;15TH WORD CONTROL MAP..1920KW
003256  000000                      .WORD   0       ;16TH WORD CONTROL MAP..2048KW
003260  000000                      .WORD   0       ;MASK FOR UNUSED PARITY CSR BITS
```

```
003262  172135                  MPR14:  172134 +1       ;PARITY STATUS REGISTER
003264  000000                          .WORD   0       ;1ST  WORD CONTROL MAP...128KW
003266  000000                          .WORD   0       ;2ND  WORD CONTROL MAP...256KW
003270  000000                          .WORD   0       ;3RD  WORD CONTROL MAP...384KW
003272  000000                          .WORD   0       ;4TH  WORD CONTROL MAP...512KW
003274  000000                          .WORD   0       ;5TH  WORD CONTROL MAP...640KW
003276  000000                          .WORD   0       ;6TH  WORD CONTROL MAP...768KW
003300  000000                          .WORD   0       ;7TH  WORD CONTROL MAP...896KW
003302  000000                          .WORD   0       ;8TH  WORD CONTROL MAP..1024KW
003304  000000                          .WORD   0       ;9TH  WORD CONTROL MAP..1152KW
003306  000000                          .WORD   0       ;10TH WORD CONTROL MAP..1280KW
003310  000000                          .WORD   0       ;11TH WORD CONTROL MAP..1408KW
003312  000000                          .WORD   0       ;12TH WORD CONTROL MAP..1536KW
003314  000000                          .WORD   0       ;13TH WORD CONTROL MAP..1664KW
003316  000000                          .WORD   0       ;14TH WORD CONTROL MAP..1792KW
003320  000000                          .WORD   0       ;15TH WORD CONTROL MAP..1920KW
003322  000000                          .WORD   0       ;16TH WORD CONTROL MAP..2048KW
003324  000000                          .WORD   0       ;MASK FOR UNUSED PARITY CSR BITS
003326  172137                  MPR15:  172136 +1       ;PARITY STATUS REGISTER
003330  000000                          .WORD   0       ;1ST  WORD CONTROL MAP...128KW
003332  000000                          .WORD   0       ;2ND  WORD CONTROL MAP...256KW
003334  000000                          .WORD   0       ;3RD  WORD CONTROL MAP...384KW
003336  000000                          .WORD   0       ;4TH  WORD CONTROL MAP...512KW
003340  000000                          .WORD   0       ;5TH  WORD CONTROL MAP...640KW
003342  000000                          .WORD   0       ;6TH  WORD CONTROL MAP...768KW
003344  000000                          .WORD   0       ;7TH  WORD CONTROL MAP...896KW
003346  000000                          .WORD   0       ;8TH  WORD CONTROL MAP..1024KW
003350  000000                          .WORD   0       ;9TH  WORD CONTROL MAP..1152KW
003352  000000                          .WORD   0       ;10TH WORD CONTROL MAP..1280KW
003354  000000                          .WORD   0       ;11TH WORD CONTROL MAP..1408KW
003356  000000                          .WORD   0       ;12TH WORD CONTROL MAP..1536KW
003360  000000                          .WORD   0       ;13TH WORD CONTROL MAP..1664KW
003362  000000                          .WORD   0       ;14TH WORD CONTROL MAP..1792KW
003364  000000                          .WORD   0       ;15TH WORD CONTROL MAP..1920KW
003366  000000                          .WORD   0       ;16TH WORD CONTROL MAP..2048KW
003370  000000                          .WORD   0               ;MASK FOR UNUSED PARITY CSR BITS
                        ;THIS IS THE END OF THE TABLE !
003372                  MPRX:   .BLKW   17.     ;TABLE TO HOLD JUST PARITY STATUS REGISTERS THAT EXIST.
                                                ; (THE EXTRA WORD IS FOR A TERMINATOR.)
```

```
                                .SBTTL  ERROR POINTER TABLE
                                ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
                                ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
                                ;*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
                                ;*NOTE1:        IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
                                ;*NOTE2:        EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
                                ;*      EM              ;;POINTS TO THE ERROR MESSAGE
                                ;*      DH              ;;POINTS TO THE DATA HEADER
                                ;*      DT              ;;POINTS TO THE DATA
                                ;*      DF              ;;POINTS TO THE DATA FORMAT
     003434                     $ERRTB:
1232 003434                     CHGG1:
1233                            ;*  ITEM 1
1234 003434  025152                     DM1             ;PARITY REGISTER DATA ERROR.
1235 003436  026512                     DH1             ;PC,REG,S/B,WAS
1236 003440  002042                     DT1             ;$ERRPC,$GDADR,$GDDAT,$BDDAT
1237 003442  027250                     DF1             ;16,22,16,16
1238                            ;*  ITEM 2
1239 003444  025206                     DM2             ;ADDRESS TEST ERROR(TST1-5).
1240 003446  026535                     DH2             ;V/PC,P/PC,MA,S/B,WAS
1241
1242 003450  002054                     DT2             ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
1243 003452  027254                     DF2             ;16,22,22,16,16
1244                            ;*  ITEM 3
1245 003454  025206                     DM2             ;ADDRESS TEST ERROR(TST1-5).
1246 003456  027214                     DH31            ;V/PC,P/PC,MA,S/B,WAS
1247 003460  002054                     DT2             ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
1248 003462  027261                     DF3             ;16,22,22,8,8
1249                            ;*  ITEM 4
1250 003464  025242                     DM4             ;CONSTANT DATA ERROR(TST6-10).
1251 003466  026535                     DH2             ;V/PC,P/PC,MA,S/B,WAS
1252 003470  002054                     DT2             ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
1253 003472  027254                     DF2             ;16,22,22,16,16
1254                            ;*  ITEM 5
1255 003474  025300                     DM5             ;ROTATING BIT ERROR(TST11-12).
1256 003476  026535                     DH2             ;V/PC,P/PC,MA,S/B,WAS
1257 003500  002054                     DT2             ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
1258 003502  027254                     DF2             ;16,22,22,16,16
1259                            ;*  ITEM 6
1260 003504  025336                     DM6             ;MOS REFRESH TEST ERROR (TST24-25).
1261 003506  026535                     DH2             ;V/PC,P/PC,MA,S/B,WAS
1262 003510  002054                     DT2             ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
1263 003512  027254                     DF2             ;16,22,22,16,16
1264                            ;*  ITEM 7
1265 003514  025402                     DM7             ;FATAL ERROR HALT
1266 003516  000000                     0
1267 003520  000000                     0
1268 003522  000000                     0
1269                            ;*  ITEM 10
1270 003524  025423                     DM10            ;MARCHING 1'S AND 0'S ERROR(TST23).
1271 003526  026535                     DH2             ;V/PC,P/PC,MA,S/B,WAS
1272 003530  002054                     DT2             ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
1273 003532  027254                     DF2             ;16,22,22,16,16
1274                            ;*  ITEM 11
1275 003534  025467                     DM11            ;PARITY MEMORY ADDRESS ERROR(TST13).
1276 003536  027214                     DH31            ;V/PC,P/PC,MA,S/B,WAS
1277 003540  002054                     DT2             ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
```

```
1278 003542  027261                    DF3             ;16,22,22,8,8
1279                            ;*  ITEM 12
1280 003544  025533                    DM12            ;DATIO WITH WRONG PARITY DIDN'T TRAP(TST13).
1281 003546  026574                    DH12            ;V/PC,P/PC,MA,S/B
1282 003550  002070                    DT12            ;$VERPC,$ERRPC,$GDADR,$GDDAT
1283 003552  027261                    DF3             ;16,22,22,8
1284                            ;*  ITEM 13
1285 003554  025607                    DM13            ;WRONG PARITY DETECTED, BUT NO REGISTER SHOWS ERROR FLAG.
1286 003556  026574                    DH12            ;V/PC,P/PC,MA,S/B
1287 003560  002070                    DT12            ;$VERPC,$ERRPC,$GDADR,$GDDAT
1288 003562  027261                    DF3             ;16,22,22,8
1289                            ;*  ITEM 14
1290 003564  025700                    DM14            ;PARITY REGISTER NOT MAPPED AS CONTROLLING THIS ADDRESS(TST13).
1291 003566  026623                    DH14            ;V/PC,P/PC,REG,MA
1292 003570  002102                    DT14            ;$VERPC,$ERRPC,$TMP0,$GDADR
1293
1294 003572  027266                    DF14            ;16,22,22,22
1295                            ;*  ITEM 15
1296 003574  025152                    DM1             ;PARITY REGISTER DATA ERROR.
1297 003576  026652                    DH15            ;V/PC,P/PC,MAUT,REG,S/B,WAS
1298 003600  002114                    DT15            ;$VERPC,$ERRPC,$GDADR,$TMP0,$GDDAT,$BDDAT
1299 003602  027266                    DF14            ;16,22,22,22,16,16
1300                            ;*  ITEM 16
1301 003604  025777                    DM16            ;MORE THAN ONE REGISTER INDICATED PARITY ERROR.
1302 003606  026623                    DH14            ;V/PC,P/PC,REG,MA
1303 003610  002102                    DT14            ;$VERPC,$ERRPC,$TMP0,$GDADR
1304 003612  027266                    DF14            ;16,22,22,22
1305                            ;*  ITEM 17
1306 003614  026056                    DM17            ;DATA SHOULDN'T HAVE CHANGED WHEN PARITY ERROR
1307                                                    ; TRAPPED(TST13).
1308 003616  027214                    DH31            ;V/PC,P/PC,MA,S/B,WAS
1309 003620  002054                    DT2             ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
1310 003622  027261                    DF3             ;16,22,22,8,8
1311                            ;*  ITEM 20
1312 003624  026154                    DM20            ;RANDOM DATA ERROR(TST14).
1313 003626  026535                    DH2             ;V/PC,P/PC,MA,S/B,WAS
1314 003630  002054                    DT2             ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
1315 003632  027254                    DF2             ;16,22,22,16,16
1316                            ;*  ITEM 21
1317 003634  026206                    DM21            ;INSTRUCTION EXECUTION ERROR(TST15 22).
1318 003636  026725                    DH21            ;V/PC,P/PC,IUT,MA,S/B,WAS
1319 003640  002132                    DT21            ;$VERPC,$ERRPC,$TMP0,$GDADR,$GDDAT,$BDDAT
1320 003642  027274                    DF21            ;16,22,16,22,16,16
1321                            ;*  ITEM 22             ;NOT USED
1322 003644  000000                    0               ;CHGG1
1323 003646  000000                    0
1324 003650  000000                    0
1325 003652  000000                    0
1326                            ;*  ITEM 23
1327 003654  026255                    DM23            ;PROGRAM CODE CHANGED WHEN RELOCATED.
1328 003656  026772                    DH23            ;V/PC,P/PC,SRC MA,DST MA,S/B,WAS
1329 003660  002150                    DT23            ;$VERPC,$ERRPC,$GDADR,$BDADR,$GDDAT,$BDDAT
1330 003662  027266                    DF14            ;16,22,22,22,16,16
1331                            ;*  ITEM 24
1332 003664  026322                    DM24            ;TRAPPED, BUT NO REGISTER HAD ERROR BIT SET.
1333 003666  027046                    DH24            ;V/PC,P/PC,TRP/PC
1334 003670  002166                    DT24            ;$VERPC,$ERRPC,$BDADR
```

```
1335 003672  027266                        DF14            ;16,22,22
1336                                ;*  ITEM 25
1337 003674  026376                        DM25            ;TRAPPED TO 114.
1338 003676  027071                        DH25            ;V/PC,P/PC,TRP/PC,REG,WAS
1339 003700  002176                        DT25            ;$VERPC,$ERRPC,$BDADR,$TMP0,$TMP1
1340 003702  027266                        DF14            ;16,22,22,22,16
1341                                ;*  ITEM 26
1342 003704  026416                        DM26            ;FAILED TO TRAP.
1343 003706  027135                        DH26            ;V/PC,P/PC,REG,WAS
1344 003710  002212                        DT26            ;$VERPC,$ERRPC,$TMP0,$TMP1
1345 003712  027254                        DF2             ;16,22,22,16
1346                                ;*  ITEM 27
1347 003714  026436                        DM27            ;(ACTION ENABLE WASN'T SET).
1348 003716  027135                        DH26            ;V/PC,P/PC,REG,WAS
1349 003720  002212                        DT26            ;$VERPC,$ERRPC,$TMP0,$TMP1
1350 003722  027254                        DF2             ;16,22,22,16
1351                                ;*  ITEM 30
1352 003724  000000                        0               ;NO MESSAGE.
1353 003726  027165                        DH30            ;REG,WAS,MA,WAS
1354 003730  002224                        DT30            ;$TMP0,$TMP1,$GDADR,$BDDAT
1355 003732  027302                        DF30            ;22,16,22,8
1356                                ;*  ITEM 31
1357 003734  026472                        DM31            ;TRAPPED TO 4
1358 003736  000000                        0               ;NO HEADER
1359 003740  002236                        DT31            ;$TMP3
1360 003742  027302                        DF30            ;22
1361
1362                                .SBTTL START:   SETUP AND MAP MEMORY
     .REPT   1
     ;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
     ;*      THIS IS THE NORMAL (SA = 200) BEGINNING OF THE PROGRAM.
     ;*      NOTE:   THIS CODE IS NOT POSITION INDEPENDENT.
     .REPT   1
     ;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*
1367 003744  105067  176006           START:  CLRB    SELFLG          ;CLEAR SELECT PARAMETER FLAG.
1368 003750  000403                           BR      STARTA          ;GO DO SETUP AND MEMORY MAP.
1369 003752  112767  177777  175776   SELECT: MOVB    #-1,    SELFLG  ;SET THE SELECT PARAMETERS FLAG.
1370 003760                           STARTA:
                                       .SBTTL  INITIALIZE THE COMMON TAGS
                                       ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
     003760  012706  001070                   MOV     #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
     003764  005026                           CLR     (R6)+           ;;CLEAR MEMORY LOCATION
     003766  022706  001130                   CMP     #SWR,R6 ;;DONE?
     003772  001374                           BNE     .-6             ;;LOOP BACK IF NO
     003774  012706  001100                   MOV     #STACK,SP       ;;SETUP THE STACK POINTER
                                       ;;INITIALIZE A FEW VECTORS
     004000  012737  000540  000024           MOV     #$PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
     004006  012737  000340  000026           MOV     #340,@#PWRVEC+2 ;;LEVEL 7
     004014  016767  006634  006624           MOV     $ENDCT,$EOPCT   ;;SETUP END-OF-PROGRAM COUNTER
                                       ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
                                       ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
     004022  013746  000004                   MOV     @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
     004026  012737  004062  000004           MOV     #30000$,@#ERRVEC        ;;SET UP ERROR VECTOR
     004034  012767  177570  175066           MOV     #DSWR,SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
     004042  012767  177570  175062           MOV     #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
     004050  022777  177777  175052           CMP     #-1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
     004056  001012                           BNE     30002$          ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
```

```
                                                        ;;AND  THE HARDWARE SWR IS NOT = -1
004060  000403                          BR      30001$          ;;BRANCH IF NO TIMEOUT
004062  012716  004070        30000$:   MOV     #30001$,(SP)    ;;SET UP FOR TRAP RETURN
004066  000002                          RTI
004070  012767  000176  175032 30001$:   MOV     #SWREG,SWR      ;;POINT TO SOFTWARE SWR
004076  012767  000174  175026          MOV     #DISPREG,DISPLAY
004104  012637  000004        30002$:   MOV     (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR
004110  005067  175066                  CLR     $PASS           ;;CLEAR PASS COUNT
004114  132767  000200  175073        BITB    #APTSIZE,$ENVM  ;;TEST USER SIZE UNDER APT
004122  001403                          BEQ     30003$          ;;YES,USE NON-APT SWITCH
004124  012767  001216  174776        MOV     #$SWREG,SWR     ;;NO,USE APT SWITCH REGISTER
004132                          30003$:
1371                            ;;*****************************************************************
1372                            ;;V DISABLE CACHE ELSE SIZING IS AFFECTED
1373                            ;;V*****************************************************************
1374 004132  016746  173646              MOV     4,-(SP)         ;;V SAVE TIME OUT VECTOR
1375 004136  016746  173644              MOV     6,-(SP)         ;;V SAVE THE STATUS
1376 004142  012767  004164  173634      MOV     #30$,4          ;;V SET UP TIME OUT VECTOR
1377 004150  005067  173632              CLR     6               ;;V CLEAR T.O. STATUS LOC
1378 004154  052737  001414  177746      BIS     #1414,@#177746  ;;V DISABLE CACHE
1379 004162  000402                      BR      31$             ;;V CACHE IS PRESENT
1380 004164  062706  000004      30$:    ADD     #4,SP           ;;V RESET STACK POINTER
1381 004170  012667  173612      31$:    MOV     (SP)+,6         ;;V RESTORE T.O. STATUS
1382 004174  012667  173604              MOV     (SP)+,4         ;;V RESTORE T.O. VECTOR
1383                            ;;VK****************************************************************
1384                            ;;VK SIZE FOR KDJ11-DA (11/53) PROCESSOR
1385                            ;;VK****************************************************************
1386 004200  005067  175574              CLR     KDJDA
1387 004204  013746  000004              MOV     @#4,-(SP)
1388 004210  012737  004244  000004      MOV     #1$,@#4
1389 004216  013700  177750              MOV     @#177750,R0
1390 004222  042700  177417              BIC     #177417,R0
1391 004226  022700  000100              CMP     #100,R0
1392 004232  001004                      BNE     1$
1393 004234  012767  177777  175536      MOV     #-1,KDJDA
1394 004242  000402                      BR      2$
1395 004244  062706  000004      1$:     ADD     #4,SP
1396 004250  012637  000004      2$:     MOV     (SP)+,@#4
1397                            ;;VK*************END KDJ11-DA SIZING**********************************
1398 004254  005067  175232              CLR     LDDISP          ;CLEAR DISPLAY REGISTER STORAGE LOCN
1399 004260  005077  174646              CLR     @DISPLAY        ;CLEAR DISPLAY REGISTER
1400                            .SBTTL  TYPE PROGRAM NAME
                                ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
004264  005227  177777                  INC     #-1             ;;FIRST TIME?
004270  001041                          BNE     30004$          ;;BRANCH IF NO
004272  022737  012722  000042          CMP     #$ENDAD,@#42    ;;ACT-11?
004300  001435                          BEQ     30004$          ;;BRANCH IF YES
004302  004567  015444                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
004306  004362                          .WORD   30005$          ;ADDRESS OF MESSAGE TO BE TYPED
                                .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
004310  005737  000042                  TST     @#42            ;;ARE WE RUNNING UNDER XXDP/ACT?
004314  001016                          BNE     30006$          ;;BRANCH IF YES
004316  126727  174672  000001          CMPB    $ENV,#1         ;;ARE WE RUNNING UNDER APT?
004324  001412                          BEQ     30006$          ;;BRANCH IF YES
004326  026727  174576  000176          CMP     SWR,#SWREG      ;;SOFTWARE SWITCH REG SELECTED?
004334  001011                          BNE     30007$          ;;BRANCH IF NO
                                ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $GTSWR ROUTINE
```

```
                                        ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
004336  106746                          MFPS    -(SP)           ;PUT THE PROCESSOR STATUS ON THE STACK
004340  105066  000001                  CLRB    1(SP)           ;HIGH BYTE CLEARED TO INSURE KERNEL MODE
                                                                ;ON PSW RETURN.
004344  004767  014310                  JSR     PC,     $GTSWR  ;GO TO THE SUBROUTINE
004350  000403                          BR      30007$
004352  112767  000001  174544  30006$: MOVB    #1,$AUTOB       ;;SET AUTO-MODE INDICATOR
004360                                  30007$:
004360  000405                          BR      30004$          ;;GET OVER THE ASCIZ
                                        ;;30005$:       .ASCIZ  <CRLF>'CVMSAB'<CRLF>
004374                                  30004$:
1401 004374  010700                    MOV     PC,     R0      ;GET CURRENT PROGRAM COUNTER.
1402 004376  022700  004376            9$:     CMP     #9$,    R0      ;CHECK IF THE PROGRAM IS RELOCATED.
1403 004402  001402                    BEQ     10$             ;BR IF PROGRAM NOT RELOCATED.
1404 004404  000167  173670            JMP     RESTAR          ;GO TRY TO RELOCTED BEFORE CONTINUING.
1405 004410  012767  000001  174116  10$:    MOV     #1,     PRGMAP  ;INITIALIZE PROGRAM MAP.
1406 004416  005067  174110            12$:    CLR     RELOCF  ;INIT THE RELOCATION FACTOR.
1407 004422  105737  001214            TSTB    @#$ENV          ;CHECK FOR APT11
1408 004426  001011                    BNE     13$             ;BR IF APT11
1409 004430  005737  000042            TST     @#42            ;CHECK FOR STANDALONE
1410 004434  001406                    BEQ     13$             ;BR IF STANDALONE
1411 004436  023737  000042  000046    CMP     @#42,@#46       ;CHECK FOR ACT11
1412 004444  001402                    BEQ     13$             ;BR IF ACT11
1413                                                            ;MUST BE XXDP
1414 004446  004767  011100            JSR     PC,SAVLDR       ;GO SAVE LOADERS
1415
1416                                    ;*  CHECK IF MEMORY MANAGEMENT IS AVAILABLE, AND SET IT UP IF IT IS.
1417                                    ;*  INITIALIZE THE MEMMAP TABLE
1418 004452  012700  001514            13$:    MOV     #MEMMAP,R0      ;LOAD R0 WITH MEMMAP TABLE ADDRESS
1419 004456  012701  000020            MOV     #16.,R1         ;LOAD COUNTER
1420 004462  005020                    14$:    CLR     (R0)+           ;CLEAR MEMMAP TABLE ENTRY
1421 004464  077102                    SOB     R1,14$          ;DECREMENT COUNTER 16. TIMES
1422                                                            ;IF COUNTER NOT = 0 THEN CLEAR NEXT ENTRY
1423
1424 004466  005067  175304            CLR     LSIFLG          ;INIT LSI-11 /2/QUAD PROCESSOR FLAG
1425 004472  012767  004504  173310    MOV     #15$,RESVEC     ;FIND OUT IF LSI-11/2/QUAD
1426 004500  000007                    MFPT                    ;MFPT INSTRUCTION WILL CAUSE TRAP
1427                                                            ;ON LSI-11/2
1428 004502  000404                    BR      16$             ;11/23 OR LATER WILL BRANCH
1429 004504  062706  000004            15$:    ADD     #4,SP           ;LSI-11 RETURN,CORRECT STACK
1430 004510  005267  175262            INC     LSIFLG          ;AND SET LSI-11/2 FLAG
1431
1432                                    ;*  CHECK IF MEMORY MANAGEMENT IS AVAILABLE, AND SET IT UP IF IT IS
1433 004514  005067  174016            16$:    CLR     MMAVA           ;CLEAR KT AND 22 BIT ADDRESSING FLAG
1434 004520  032777  010000  174402    BIT     #SW12,@SWR      ;CHECK FOR INHIBIT KT11 SWITCH
1435 004526  001047                    BNE     NONKT           ;BRANCH IF SET
1436 004530  012737  004646  000004    MOV     #NONKT,@#ERRVEC ;SET UP TIMEOUT TRAP VECTOR
1437 004536  005037  177572            CLR     @#SR0           ;CLEAR MEM MGMT STATUS REGISTER
1438 004542  004767  006232            JSR     PC,MMINIT       ;MEM MGMT INITIALIZATION ROUTINE
1439 004546  005267  173764            INC     MMAVA           ;SET MEM MGMT AVAILABLE FLAG
1440 004552  004567  015174            JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
     004556  023650                    .WORD   MMAMES          ;ADDRESS OF MESSAGE TO BE TYPED
                                                                ;"KT11 AVAILABLE"
1441
1442
1443                                    ;*  CHECK IF 22 BIT SYSTEM AVAILABLE AND SET IT UP IF IT IS
1444 004560  012737  004630  000004    MOV     #22$,@#ERRVEC   ;SET UP FOR TIME OUT VECTOR
```

```
                                        CLR     @#0                     ;CLEAR LOCATION 0
                                        MOV     #10000,@#KIPAR2         ;SET PAR2 TO LOC 128K + 2
                                        BIS     #BIT4,@#SR3             ;TURN ON 22 BIT ADDRESSING
                                        MOV     #-1,@#40000             ;NOW WRITE TO LOC 128K + 2
                                        TST     @#0                     ;IF LOC 0 = 0
                                        BEQ     22$                     ;  THEN 22 BIT SYSTEM
                                        CLR     @#SR3                   ;  ELSE 18 BIT SYSTEM, DISABLE 22 BIT ADR
                                        BR      KTSIZ                   ;       AND GO SIZE MEMORY

                                ;*  TIME OUT TRAP TO HERE OR MEMORY EXISTS AT 128K + 2
                                22$:    BIS     #BIT15,MMAVA            ;  ELSE SET 22 BIT FLAG
                                        JSR     R5,     $PRINT          ;GO PRINT OUT THE FOLLOWING MESSAGE.
                                        .WORD   AVAL22                  ;ADDRESS OF MESSAGE TO BE TYPED
                                                                        ;"22 BIT ADR AVAILABLE"
                                        BR      KTSIZ                   ;GO SIZE MEMORY


                                ;;*****************************************************************
                                ;*  THIS ROUTINE WILL MAP MEMORY IN 8K SEGMENTS.  SUPPORTS ONLY THE SIZING
                                ;*  OF 16 BIT ADDRESSING  WITHOUT MEM MGMT SUPPORT.
                                ;*      STORAGE USED:
                                ;*              R0      = MEMMAP POINTER ... LO 128K
                                ;*              R2      = ADDRESS POINTER
                                ;*              R3      = BANK POINTER ... LO 128K
                                ;*              FLG30K  = 30K MEMORY FLAG
                                ;*      LITERALS:
                                ;*              MASK8K  = 37777
                                ;;*****************************************************************
                                NONKT:  MOV     #STACK,SP               ;SET-UP THE STACK
                                        MOV     #MEMMAP,R0              ;SET UP MEMORY MAP PTR TO LO 128K
                                        CLR     R2                      ;SET ADDRESS PTR TO 0
                                        MOV     #1,R3                   ;SET UP 8K BANK POINTER
                                        MOV     #2$,@#ERRVEC            ;SET UP TIME OUT VECTOR
                                1$:     MOV     (R2),(R2)+              ;READ AND WRITE ALL MEMORY
                                        BIT     #MASK8K,R2              ;IF NOT 8K BOUNDARY
                                        BNE     1$                      ;  THEN CHECK NEXT LOCATION
                                        BIS     R3,(R0)                 ;  ELSE SET BANK FLAG IN MEMMAP
                                        BR      3$                      ;       AND DO SOME MORE

                                ;*  TIMEOUT TRAPS TO HERE
                                2$.     ADD     #4,SP                   ;RESTORE STACK POINTER
                                        CMP     #160000,R2              ;IF NOT 28K BOUNDARY
                                        BNE     20$                     ;  THEN BRANCH
                                        BR      21$                     ;  ELSE SET UP POINTERS
                                20$:    CMP     #170000,R2              ;IF NOT 30K BOUNDARY
                                        BNE     22$                     ;  THEN BRANCH
                                        INC     FLG30K                  ;  ELSE SET 30K MEMORY FLAG
                                21$:    BIS     R3,(R0)                 ;SET BANK FLAG IN MEMMAP
                                        BR      4$                      ;BRANCH ALL DONE
                                22$:    BIS     #MASK8K,R2              ;POINT TO LAST ADDRESS OF 8K BANK
                                        INC     R2                      ;POINT TO 1ST ADDRESS OF NEXT BANK

                                3$:     ASLB    R3                      ;UPDATE BANK POINTER
                                        BIT     #BIT4,R3                ;IF NOT DONE WITH 32K
                                        BEQ     1$                      ;  THEN TRY SOME MORE
                                4$:     BR      DISMAP                  ;GO TYPE OUT MAP
```

```
1500
1501                                    ;;*****************************************************************
1502                                    ;*  THIS ROUTINE WILL MAP MEMORY IN 8K SEGMENTS.  MEMORY MANAGEMENT REGISTERS
1503                                    ;*  KIPAR2 AND KIPAR3 ARE USED TO MAP THE 8K BANKS OF MEMORY.
1504                                    ;*  IF MEMORY EXISTS NEXT TO THE I/O PAGE (I.E. 760000 OR 17760000)
1505                                    ;*  THEN THE LAST BANK WILL BE ACKNOWLEDGED AS EXISTING.
1506                                    ;*      STORAGE USED:
1507                                    ;*              R0      = MEMMAP POINTER
1508                                    ;*              R2      - ADDRESS POINTER
1509                                    ;*              R3      = BANK POINTER
1510                                    ;*              KIPAR2  = MAPPED TO 1ST 4K OF PRESENT 8K BANK
1511                                    ;*              KIPAR3  = MAPPED TO 2ND 4K OF PRESENT 8K BANK
1512                                    ;*      LITERALS USED:
1513                                    ;*              MASK8K  = MASK OF 8K (37777)
1514                                    ;;*****************************************************************
1515 004760  012706  001100          KTSIZ:  MOV     #STACK,SP       ;SET-UP THE STACK
1516 004764  012700  001514                  MOV     #MEMMAP,R0      ;SET-UP MEMMAP PTR TO FIRST ENTRY
1517 004770  012702  040000                  MOV     #40000,R2       ;INIT VIRTUAL ADDRESS TO 0 MAPPED THRU PAR2
1518 004774  005037  172344                  CLR     @#KIPAR2        ;INIT PAR2 TO LOC 0
1519 005000  012737  000200  172346          MOV     #200,@#KIPAR3   ;SET PAR3 TO 2ND 4K BANK
1520 005006  012737  005034  000004          MOV     #3$,@#ERRVEC    ;LOAD TIME OUT VECTOR
1521
1522 005014  012703  000001          1$:     MOV     #BIT0,R3        ;SET-UP 8K BANK POINTER
1523
1524 005020  011222                  2$:     MOV     (R2),(R2)+      ;READ AND WRITE ALL MEMORY
1525 005022  032702  037777                  BIT     #MASK8K,R2      ;IF NOT 8K BOUNDARY
1526 005026  001374                          BNE     2$              ;  THEN TRY SOMEMORE
1527 005030  050310                          BIS     R3,(R0)         ;  ELSE SET BANK FLAG IN MEMMAP
1528 005032  000424                          BR      5$              ;AND GO UPDATE VARIABLES AND CONTINUE
1529
1530                                    ;*  TIMEOUT TRAPS TO HERE
1531 005034  062706  000004          3$:     ADD     #4,SP           ;RESTORE STACK POINTER
1532 005040  022702  060000                  CMP     #60000,R2       ;IF NOT POSSIBLY THE I/O PAGE
1533 005044  001017                          BNE     5$              ;  THEN GO TEST SOMEMORE
1534 005046  005767  173464                  TST     MMAVA           ;  ELSE IF 22 BIT ADDRESSING
1535 005052  100406                          BMI     4$              ;    THEN GO SEE IF 2M I/O PAGE
1536 005054  022737  007600  172346          CMP     #7600,@#KIPAR3  ;    ELSE IF NOT I/O BOUNDARY FOR 18 BITS
1537 005062  001010                          BNE     5$              ;      THEN GO UPDATE VARIABLES AND TRY SOMEMORE
1538 005064  050310                          BIS     R3,(R0)         ;      ELSE SET BANK EXISTS IN MEMMAP
1539 005066  000433                          BR      7$              ;AND GO TYPE MEMORY MAP
1540
1541 005070  022737  177600  172346  4$:     CMP     #177600,@#KIPAR3        ;IF NOT 2M I/O BOUNDARY
1542 005076  001002                          BNE     5$              ;  THEN GO TRY SOMEMORE SIZING
1543 005100  050310                          BIS     R3,(R0)         ;  ELSE SET BANK IN MEMMAP
1544 005102  000425                          BR      7$              ;AND GO TYPE MEMORY MAP
1545
1546 005104  062737  000400  172344  5$:     ADD     #400,@#KIPAR2   ;UPDATE MAP TO NEXT
1547 005112  062737  000400  172346          ADD     #400,@#KIPAR3   ;    8K BANK
1548 005120  012702  040000                  MOV     #40000,R2       ;RESTORE ADDRESS POINTER TO 1ST ADDRESS OF THIS BANK
1549 005124  005767  173406                  TST     MMAVA           ;IF NOT 18 BIT ADDR
1550 005130  100403                          BMI     6$              ;THEN GO TEST SOMEMORE
1551 005132  006303                          ASL     R3              ;18 BIT ADDR. = 1 WORD
1552 005134  001331                          BNE     2$              ;IF NOT END OF 18 BIT ADDR
1553                                                                 ;  THEN GO SIZE SOMEMORE
1554 005136  000407                          BR      7$              ;  ELSE ALL DONE 18 BIT SIZING.
1555 005140  006303                  6$:     ASL     R3              ;UPDATE BANK POINTER
1556 005142  001326                          BNE     2$              ;IF NOT END OF THIS MEMMAP ENTRY, THEN CONTINUE
```

```
1557 005144  062700  000002                ADD     #2,R0           ;  ELSE UPDATE TO NEX MEMMAP ENTRY
1558 005150  022700  001554                CMP     #MEMMAP+40,R0   ;IF NOT END OF MEMMAP TABLE
1559 005154  001317                        BNE     1$              ;    THEN GO SIZE SOMEMORE
1560                                                               ;    ELSE ALL DONE SIZING
1561 005156  000400                7$:     BR      DISMAP          ;GO TYPE OUT MEMORY MAP
1562
1563
1564                                       ;;*****************************************************************
1565                                       ;*  ROUTINE WILL TYPE OUT MEMMAP, LOAD TEST MAP (SAVTST) AND CHECKS
1566                                       ;*  TO INSURE LOWEST 16K OF MEMORY IS AVAILABLE FOR TEST TO RUN
1567                                       ;*      STORAGE LOCATIONS:
1568                                       ;*              R0      = MEMMAP POINTER ... LO 128K
1569                                       ;*              R1      = COUNTER
1570                                       ;*              R2      = SAVTST POINTER ... LO 128K
1571                                       ;;*****************************************************************
1572 005160  012700  001514          DISMAP: MOV     #MEMMAP,R0      ;LOAD R0 WITH MEMMAP ADR
1573 005164  012737  023430  000004        MOV     #ERRTRP,@#ERRVEC;SET UP TIME OUT VECTOR
1574 005172  004567  014554                JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
     005176  023742                        .WORD   MEMMES          ;ADDRESS OF MESSAGE TO BE TYPED
                                                                   ;"MEMORY MAP:"
1575 005200  004767  011622                JSR     PC,TYPMAP       ;GO TYPE THE MAP
1576 005204  004567  014542                JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
     005210  001171                        .WORD   $CRLF           ;ADDRESS OF MESSAGE TO BE TYPED
1577 005212  012702  001614                MOV     #SAVTST,R2      ;LOAD ADR OF SAVTST TABLE TO BE CLEARED
1578 005216  012700  001514                MOV     #MEMMAP,R0      ;LOAD ADR OF MEMMAP TABLE
1579 005222  012701  000020                MOV     #16.,R1         ;LOAD COUNTER
1580 005226  005012                1$:     CLR     (R2)            ;CLEAR SAVTST TABLE ENTRY
1581 005230  012022                        MOV     (R0)+,(R2)+     ;LOAD SAVTST FROM MEMMAP
1582 005232  077103                        SOB     R1,1$           ;DECREMENT CTR 16 TIMES
1583 005234  016700  174254                MOV     MEMMAP,R0       ;LOAD R0 WITH MAP OF 1ST 128K
1584 005240  042700  177774                BIC     #177774,R0      ;MASK ALL BUT BOTTOM 16K
1585 005244  022700  000003                CMP     #3,R0           ;IF BOTTOM 16K IS ALL THERE
1586 005250  001404                        BEQ     GMPR            ;  THEN GO RUN
1587 005252  004567  014474                JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
     005256  024022                        .WORD   INSUFF          ;ADDRESS OF MESSAGE TO BE TYPED
                                                                   ;"FIRST 16K OF MEMORY NOT ALL THERE!"
1588 005260  000000                        HALT                    ;FATAL ERROR HALT ...
1589                                                               ;MEMORY IS NOT CONFIGURED TO RUN THIS PROGRAM
1590
1591
1592                                   .SBTTL  MAP PARITY REGISTERS
1593                                   ;;*****************************************************************
1594                                   ;*  SEARCH FOR PARITY REGISTERS PRESENT AND TYPE ADDRESSES OF THOSE FOUND
1595                                   ;*  THAT ARE FUNCTIONAL AND HAVE CORRESPONDING PARITY MEMORY
1596                                   ;;*****************************************************************
1597
1598 005262  012704  003372          GMPR:   MOV     #MPRX,  R4      ;SET UP POINTER TO PARITY REG EXIST TABLE.
1599 005266  032777  000100  173634        BIT     #SW06,  @SWR    ;CHECK FOR INHIBIT PARITY SWITCH.
1600 005274  001040                        BNE     GMPRD           ;BR IF INHIBIT PARITY.
1601 005276  012703  002272                MOV     #MPR0,  R3      ;SET UP TABLE POINTER
1602 005302  012737  005324  000004        MOV     #GMPRB, @#ERRVEC ;SET UP TIMEOUT TRAP SERVICE
1603 005310  042713  000001          GMPRA:  BIC     #1,     (R3)    ;CLEAR FLAG BIT IN TABLE
1604 005314  005773  000000                TST     @(R3)           ;DOES THIS MEMORY PARITY REGISTER EXIST.
1605                                   ;*  IF IT DOESN'T EXIST, A TIMEOUT TRAP WILL GO TO "GMPRB".
1606 005320  012324                        MOV     (R3)+,  (R4)+   ;SAVE IT IN THE PARITY REG EXIST TABLE.
1607 005322  000404                        BR      GMPRBA          ;SKIP TIMEOUT SERVICE CODE
1608                                   ;*  TIMEOUT COMES HERE
```

N3


```
 1609 005324  062706  000004          GMPRB:  ADD     #4,SP           ;RESTORE STACK POINTER
 1610 005330  052723  000001                  BIS     #1,     (R3)+   ;SET FLAG TO INDICATE REGISTER NOT PRESENT
 1611 005334  012701  000021          GMPRBA: MOV     #17.,R1         ;LOAD COUNTER
 1612 005340  005023                  GMPRC:  CLR     (R3)+           ;CLEAR ENTRY IN THIS GPR TABLE
 1613 005342  077102                          SOB     R1,GMPRC        ;DECREMENT COUNTER AND CONTINUE
 1614                                                                 ;CLEARING COUNTER UNTIL COMPLETED.
 1615 005344  020327  003372                  CMP     R3,     #MPRX   ;HAVE WE CHECKED ALL REGISTERS?
 1616
 1617 005350  103757                          BLO     GMPRA           ;NO   GO BACK TO CHECK NEXT ONE
 1618 005352  005014                          CLR     (R4)            ;SET TERMINATOR IN PARITY REG EXIST TABLE.
 1619 005354  012737  023430  000004          MOV     #ERRTRP,@#ERRVEC ; RESTORE TRAPCATCHER
 1620 005362  005767  176004                  TST     MPRX                    ;ANY PARITY REGISTERS PRESENT?
 1621 005366  001006                          BNE     MPAMEM                  ;YES - GO TEST CONTROLS PRESENT
 1622 005370  004567  014356                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
      005374  024103                          .WORD   MTR             ;ADDRESS OF MESSAGE TO BE TYPED
                                                                     ;"NO MEMORY PARITY REGISTERS FOUND"
 1623 005376  005014                  GMPRD:  CLR     (R4)            ;MAKE SURE TABLE IS CLEAR.
 1624 005400  000167  000744                  JMP     MANUAL                  ;AND SKIP ALL CONTROLS TESTING
 1625
```

```
                                .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
                                ;;*****************************************************************
005404  004767  011262        MPAMEM: JSR     PC,CLRPAR       ;INITIALIZE ALL PARITY REGISTERS
005410  012702  014000                MOV     #14000,R2       ;SET ADDRESS TO 14000 TO WRITE WRONG PARITY
005414  005767  173116                TST     MMAVA           ;IF NO MEM MGMT
005420  001404                        BEQ     MAPRB           ;  THEN GO MAP PARITY MEMORY
005422  012702  054000                MOV     #54000,R2       ;  ELSE SET ADDRESS POINTER TO MAP THRU PAR2
005426  004767  005346                JSR     PC,MMINIT       ;SET UP MEM MGMT REGISTERS

005432  012703  001654        MAPRB:  MOV     #PMEMAP,R3      ;LOAD PMEMAP TABLE ADR
005436  012704  000020                MOV     #16.,R4         ;LOAD COUNTER
005442  005023                1$:     CLR     (R3)+           ;CLEAR ALL OF TABLE
005444  077402                        SOB     R4,1$           ;IF NOT DONE CLEARING THEN TRY SOMEMORE

005446  004767  011220                JSR     PC,CLRPAR       ;GO INITIALIZE ALL PARITY REGISTERS
005452  005000                        CLR     R0              ;INIT INDEX FOR MEMMAP AND PMEMAP TABLES
005454  012705  000002                MOV     #2,R5           ;INIT INDEX FOR MPR TABLES
005460  012701  000001                MOV     #BIT0,R1        ;INIT BANK POINTER

005464  012703  002272        2$:     MOV     #MPR0,R3        ;INIT MPR  TABLE ADDRESS POINTER
005470  010304                3$:     MOV     R3,R4           ;UPDATE TO NEW TABLE
005472  060504                        ADD     R5,R4           ;UPDATE INDEX THRU NEW TABLE

005474  032713  000001                BIT     #BIT0,(R3)      ;IF CSR IS NOT PRESENT
005500  001021                        BNE     4$              ;  THEN GO TRY AGAIN
005502  013773  001766  000000        MOV     @#WWP,@(R3)     ;  ELSE SET WRITE WRONG PARITY BIT
005510  011212                        MOV     (R2),(R2)       ;WRITE WRONG PARITY
005512  005712                        TST     (R2)            ;READ WRONG PARITY
005514  043773  001766  000000        BIC     @#WWP,@(R3)     ;CLEAR WRITE WRONG PARITY BIT
005522  005773  000000                TST     @(R3)           ;IF NO PARITY ERROR
005526  100006                        BPL     4$              ;  THEN REGISTER DOES NOT CONTROL THIS MEMORY
005530  012763  070032  000042        MOV     #70032,42(R3)   ;  ELSE SAVE THE PARITY MASK IN TABLE
005536  050160  001654                BIS     R1,PMEMAP(R0)   ;SET BANK IN PMEMAP
005542  050114                        BIS     R1,(R4)         ;SET BANK IN MPR TABLES
005544  062703  000044        4$:     ADD     #44,R3          ;UPDATE TO  NEXT MPR TABLE
005550  022703  003372                CMP     #MPRX,R3        ; IF NOT END OF TABLE
005554  101345                        BHI     3$              ;  THEN TRY SOMEMORE
```

```
 1684 005556  011212                        MOV     (R2),(R2)       ;  ELSE CLEAR BAD PARITY
 1685
 1686 005560  005767  172752          5$:    TST     MMAVA           ;IF NO MEM MGMT
 1687 005564  001417                        BEQ     6$              ;  THEN GO UPDATE FOR 16 BIT SYSTEM
 1688 005566  062737  000400  172344        ADD     #400,@#KIPAR2   ;  ELSE UPDATE PAR2 TO NEXT 8K BANK
 1689 005574  006301                        ASL     R1              ;UPDATE BANK POINTER TO NEXT BANK
 1690 005576  001020                        BNE     7$              ;IF STILL SOME TO CHECK IN THIS 128K BANK THEN DO IT
 1691 005600  062700  000002                ADD     #2,R0           ;  ELSE UPDATE INDEX FOR MEMMAP AND PMEMAP TABLES
 1692 005604  062705  000002                ADD     #2,R5           ;UPDATE INDEX FOR GPR TABLES
 1693 005610  012701  000001                MOV     #BIT0,R1        ;INIT BANK POINTER
 1694 005614  022700  000040                CMP     #40,R0          ;IF END OF MEMMAP
 1695 005620  001413                        BEQ     9$              ; THEN GO TYPE MEM PARITY MAPS
 1696 005622  000406                        BR      7$              ;GO TRY SOMEMORE
 1697
 1698 005624  062702  040000          6$:    ADD     #40000,R2       ;UPDATE ADDRESS TO NEXT 8K BANK
 1699 005630  106301                        ASLB    R1              ;UPDATE BANK POINTR
 1700 005632  032701  000020                BIT     #BIT4,R1        ;IF DONE WITH 16 BITS
 1701 005636  001004                        BNE     9$              ;  THEN FINISHED
 1702
 1703 005640  036001  001514          7$:    BIT     MEMMAP(R0),R1   ;IF BANK DOES NOT EXIST
 1704 005644  001745                        BEQ     5$              ;  THEN GET ANOTHER BANK
 1705
 1706 005646  000706                        BR      2$              ;  THEN GO DO SOMEMORE
 1707
 1708 005650  000167  000000          9$:    JMP     TMAP            ;GO TYPE PARITY MAPS
 1709
 1710                                .SBTTL  DISPLAY PARITY MEMORY MAP
 1711                                ;;*****************************************************************
 1712                                ;*  ROUTINE TO TYPE MAP OF WHERE PARITY MEMORY IS PRESENT AND WHICH
 1713                                ;*  CONTROL REGISTERS CONTROL WHICH MEMORY.
 1714                                ;*      STORAGE USED:
 1715                                ;*              R0      = FIRST ADDRESS OF MAP TO BE TYPED
 1716                                ;*              R1      = PARITY REGISTER ADDRESS ... BITS 14-0
 1717                                ;*              R2      = PARITY REGISTER ADDRESS ... BITS 21-15
 1718                                ;*              R3      = MPR TABLE ENTRY
 1719                                ;;*****************************************************************
 1720 005654  004767  011012        TMAP:   JSR     PC,CLRPAR       ;INITIALIZE ALL PARITY REGISTERS PRESENT
 1721 005660  004567  014066                JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
      005664  023760                        .WORD   MTMAP           ;ADDRESS OF MESSAGE TO BE TYPED
                                                                    ;"PARITY MEMORY MAP:"
 1722 005666  012703  002272                MOV     #MPR0,R3        ;INIT MPR TABLE POINTER
 1723
 1724 005672  032713  000001        1$:     BIT     #BIT0,(R3)      ;IF THIS REGISTER IS NOT PRESENT
 1725 005676  001050                        BNE     6$              ;  THEN GO TRY AGAIN
 1726
 1727 005700                        2$:
      005700  004567  014046                JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
      005704  024367                        .WORD   MX1             ;ADDRESS OF MESSAGE TO BE TYPED
                                                                    ;"PARITY REGISTER AT"
 1728 005706  011301                        MOV     (R3),R1         ;SAVE PARITY REGISTER ADDRESS
 1729 005710  042701  100000                BIC     #100000,R1      ;DEVELOP BITS 14-0 OF REGISTER ADDRESS
 1730 005714  005767  172616                TST     MMAVA           ;IF TYPE OF MEMORY MANAGEMENT
 1731 005720  100404                        BMI     3$              ;  THEN 22 BIT BRANCH
 1732 005722  001006                        BNE     4$              ;  OR 18 BIT BRANCH
 1733 005724  012702  000001                MOV     #1,R2           ;  ELSE 16 BIT SET BITS 21-15 FOR PRINT OUT
 1734 005730  000405                        BR      5$              ;AND GO TYPE OUT
 1735
```

```
 1736 005732  012702  000177          3$:     MOV     #177,R2         ;LOAD BITS 21-15 FOR PRINT OUT
 1737 005736  000402                          BR      5$              ;AND GO TYPE OUT
 1738
 1739 005740  012702  000007          4$:     MOV     #7,R2           ;LOAD BITS 21-15 FOR PRINT OUT
 1740
 1741 005744                          5$:
      005744  010246                          MOV     R2,-(SP)        ;;SAVE R2 FOR TYPEOUT
                                               ;;TYPE ADDRESS BITS 21-15
                                      ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOS ROUTINE
                                      ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
      005746  106746                          MFPS    -(SP)           ;PUT THE PROCESSOR STATUS ON THE STACK
      005750  105066  000001                  CLRB    1(SP)           ;HIGH BYTE CLEARED TO INSURE KERNEL MODE
                                               ;ON PSW RETURN.
      005754  004767  015220                  JSR     PC,     $TYPOS  ;GO TO THE SUBROUTINE
      005760     003                          .BYTE   3               ;;TYPE 3 DIGIT(S)
      005761     000                          .BYTE   0               ;;SUPPRESS LEADING ZEROS
 1742 005762  010146                          MOV     R1,-(SP)        ;;SAVE R1 FOR TYPEOUT
                                               ;;TYPE ADDRESS BITS 14-0
                                      ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOS ROUTINE
                                      ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
      005764  106746                          MFPS    -(SP)           ;PUT THE PROCESSOR STATUS ON THE STACK
      005766  105066  000001                  CLRB    1(SP)           ;HIGH BYTE CLEARED TO INSURE KERNEL MODE
                                               ;ON PSW RETURN.
      005772  004767  015202                  JSR     PC,     $TYPOS  ;GO TO THE SUBROUTINE
      005776     005                          .BYTE   5               ;;TYPE 5 DIGIT(S)
      005777     001                          .BYTE   1               ;;TYPE LEADING ZEROS
 1743 006000  004567  013746                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
      006004  024415                          .WORD   MX2             ;ADDRESS OF MESSAGE TO BE TYPED
                                               ;"CONTROLS"
 1744 006006  010300                          MOV     R3,R0           ;SET UP R0 FOR TYPMAP ROUTINE
 1745 006010  062700  000002                  ADD     #2,R0           ;POINT TO MAP ENTRY FOR MPR TABLE
 1746 006014  004767  011006                  JSR     PC,TYPMAP       ;GO TYPE MEMORY COVERED BY THIS REGISTER
 1747
 1748 006020  062703  000044          6$:     ADD     #44,R3          ;UPDATE TO NEXT REGISTER IN TABLE
 1749 006024  022703  003372                  CMP     #MPRX,R3        ;IF NOT END OF MPR TABLE
 1750 006030  101320                          BHI     1$              ;  THEN DO SOMEMORE
 1751
 1752 006032  012700  000010                  MOV     #10,R0          ;LOAD DELAY
 1753 006036  012701  177777          7$:     MOV     #-1,R1          ;LOAD DELAY
 1754 006042  077101                  8$:     SOB     R1,8$           ;ALLOW DELAY TO INSURE PRINT OUT IS
 1755 006044  077004                          SOB     R0,7$           ;COMPLETED BEFORE RESET OCCURS
 1756
 1757 006046  005737  003372                  TST     @#MPRX          ;IF PARITY REGISTERS TO TEST
 1758 006052  001002                          BNE     CTRLS           ;  THEN GO TEST
 1759 006054  000167  000270                  JMP     MANUAL          ;  ELSE JUMP OVER TESTS
 1760
 1761
 1762
 1763                                  .SBTTL  TEST PARITY REGISTERS
 1764                                  ;;*****************************************************************
 1765                                  ;* SHOW THAT BITS 0, 2, 5 - 11 AND 15 OF EACH PARITY REGISTER PRESENT
 1766                                  ;*  CAN BE SET AND CLEARED.
 1767                                  ;*  THIS IS A ONCE ONLY TEST.
 1768                                  ;;*****************************************************************
 1769
 1770 006060  012703  002272          CTRLS:  MOV     #MPR0,  R3      ;LOAD INITIAL TABLE ADDRESS FOR A POINTER
 1771 006064  011302                  1$:     MOV     (R3),R2         ;GET CSR ADDRESS INTO R2
```

```
 1772 006066  032702  000001            BIT     #BIT0,R2        ;  ELSE IF CSR DOES NOT EXIST
 1773 006072  001052                    BNE     105$            ;    THEN BRANCH
 1774 006074  016367  000042  173404    MOV     42(R3),RESRVD   ;   ELSE LOAD MASK, IF EQ TO 0
 1775 006102  001446                    BEQ     105$            ;     THEN DO NOT TEST
 1776 006104  012700  000001            MOV     #1,R0           ;LOAD R0 WITH 1ST BIT TO BE TESTED
 1777 006110  005012                    CLR     (R2)            ;INITIALIZE THE PARITY REGISTER
 1778 006112  011201                    MOV     (R2),R1         ;READ THE CONTENTS OF THE PARITY REGISTER
 1779 006114  046701  173366            BIC     R_SRVD,R1       ;CLEAR RESERVED BITS, IF EQ 0
 1780 006120  001405                    BEQ     2$              ;  THEN BRANCH
 1781 006122  004767  010566  30008$:   JSR     PC,     SPRNT   ;SET UP VALUES FOR ERROR PRINTING.
      006126  004767  011712            JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
      006132  000001                    .WORD   1               ;ERROR TYPE CODE.
 1782 006134  030067  173346  2$:      BIT     R0,RESRVD       ;IF THIS BIT IS RESERVED
 1783 006140  001025                    BNE     3$              ;  THEN BRANCH AND DON'T TEST
 1784 006142  010012                    MOV     R0,(R2)         ;  ELSE SET THIS BIT IN CSR
 1785 006144  011201                    MOV     (R2),R1         ;READ AND SAVE CONTENTS OF CSR
 1786 006146  005012                    CLR     (R2)            ;CLEAR THE CSR
 1787 006150  046701  173332  106$:     BIC     RESRVD,R1       ;CLEAR RESERVED BITS
 1788 006154  020001                    CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
      006156  001405                    BEQ     30010$          ;BRANCH OVER ERROR CALL IF GOOD DATA.
      006160  004767  010572  30009$:   JSR     PC,     SPRNTO  ;SET UP VALUES FOR ERROR PRINTING.
      006164  004767  011654            JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
      006170  000001                    .WORD   1               ;ERROR TYPE CODE.
      006172                    30010$:
 1789                                                           ;MAKE SURE BIT WAS CLEARED OUT OF CSR
 1790 006172  011201                    MOV     (R2),R1         ;READ THE CONTENTS OF THE PARITY REGISTER
 1791 006174  046701  173306            BIC     RESRVD, R1      ;CLEAR BITS WHICH ARE RESERVED
 1792 006200  001405                    BEQ     3$              ;CHECK OTHER BITS - BRANCH IF OK
 1793 006202  004767  010506  30011$:   JSR     PC,     SPRNT   ;SET UP VALUES FOR ERROR PRINTING.
      006206  004767  011632            JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
      006212  000001                    .WORD   1               ;ERROR TYPE CODE.
 1794 006214  006300            3$:     ASL     R0              ;ROTATE TO GET NEXT BIT TO BE TESTED
 1795 006216  001346                    BNE     2$              ;BRANCH IF NOT DONE WITH ALL BITS
 1796 006220  062703  000044  105$:     ADD     #44,R3          ;UPDATE PTR TO NEXT ENTRY
 1797 006224  022703  003372            CMP     #MPRX,R3        ;IF NOT DONE WITH TABLE
 1798 006230  003315                    BGT     1$              ;THEN TRY AGAIN
 1799
 1800                                   ;;*****************************************************************
 1801                                   ;* SHOW THAT RESET CLEARS BITS 0,2 AND 15 OF EACH PARITY REGISTER PRESENT.
 1802                                   ;* ALSO BIT 14 IN PARITY CSR IF MEMORY IS SET FOR 22 BIT ADDRESSING
 1803                                   ;*  THIS IS A ONCE ONLY TEST.
 1804                                   ;;*****************************************************************
 1805
 1806 006232  012704  002272  RESCHK:  MOV     #MPR0,  R4      ;LOAD INITIAL TABLE ADDRESS FOR A POINTER
 1807 006236  022704  003372  1$:      CMP     #MPRX,R4        ;IF END OF TABLE
 1808 006242  003411                    BLE     100$            ;     THEN BRANCH
 1809 006244  032714  000001            BIT     #BIT0,(R4)      ;     ELSE IF CSR DOES NOT EXIST
 1810 006250  001003                    BNE     101$            ;      THEN BRANCH
 1811 006252  012774  177777  000000    MOV     #-1,@(R4)       ;      ELSE LOAD CSR WITH ALL 1 S
 1812 006260  062704  000044  101$:     ADD     #44,R4          ;UPDATE POINTER
 1813 006264  000764                    BR      1$              ;TRY AGAIN
 1814 006266  000005  100$:     RESET                           ;RESET THE WORLD
 1815 006270  012702  002272            MOV     #MPR0,R2        ;LOAD INITIAL ADDRESS FOR POINTER
 1816 006274  022702  003372  2$:      CMP     #MPRX,R2        ;IF END OF TABLE
 1817 006300  003423                    BLE     MANUAL          ;     THEN BRANCH
 1818 006302  032712  000001            BIT     #BIT0,(R2)      ;     ELSE IF CSR DOES NOT EXIST
 1819 006306  001015                    BNE     30013$          ;;V     THEN BRANCH
```

```
                                        MOV     @(R2),R1        ;       ELSE SAVE CONTENTS OF CSR
                                        CLR     @(R2)           ;CLEAR THE CSR
                                        BIC     #37772,R1       ;TEST FOR BIT14 TOO
                                                                ;18 BIT MODE = BIT14 ALWAYS READ AS ZERO
                                                                ;22 BIT MODE - BIT14 R/W,CLEARED BY RESET
                                4$:     TST     R1              ;CHECK IF REST  WERE CLEARED BY  RESET
                                        BEQ     30013$          ;BRANCH OVER ERROR CALL IF GOOD DATA.
006330  004767  010360          30012$: JSR     PC,     SPRNT   ;SET UP VALUES FOR ERROR PRINTING.
006334  004767  011504                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
006340  000001                          .WORD   1               ;ERROR TYPE CODE.
006342                          30013$:
006342  062702  000044                  ADD     #44,R2          ;UPDATE POINTER
006346  000752                          BR      2$              ;BRANCH BACK TO CHECK NEXT REGISTER

006350  005067  173406          MANUAL: CLR     FSTADR          ;INIT FIRST ADDRESS
006354  105767  173376                  TSTB    SELFLG          ;CHECK FOR SELECT PARAMETER SETUP
006360  001002                          BNE     MANUL1          ;IF FLAG SET GET USERS PARAMETERS
006362  000167  000402                  JMP     MANUL2          ;ELSE USE DEFAULT DATA
006366                          MANUL1:
006366  004567  013360                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
006372  024453                          .WORD   FADMES          ;ADDRESS OF MESSAGE TO BE TYPED
                                                                ;"FIRST ADDRESS:"
                                                                ;FIRST ADDRESS 8K BOUNDARY

                                ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $RDOCT ROUTINE
                                ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
006374  106746                          MFPS    -(SP)           ;PUT THE PROCESSOR STATUS ON THE STACK
006376  105066  000001                  CLRB    1(SP)           ;HIGH BYTE CLEARED TO INSURE KERNEL MODE
                                                                ;ON PSW RETURN.
006402  004767  013170                  JSR     PC,     $RDOCT  ;GO TO THE SUBROUTINE
006406  032716  037777                  BIT     #37777,(SP)     ;ON 8K BOUNDARY?
006412  001365                          BNE     MANUL1          ;IF NO,REASK
006414  016703  013330                  MOV     $HIOCT,R3       ;SAVE ORIGINAL HIGH BITS
006420  011604                          MOV     (SP),R4         ;AND LOW BITS
006422  006267  013322                  ASR     $HIOCT          ;DIVIDE HIGH ADDRESS INTO
006426  006016                          ROR     (SP)            ;NUMBER OF 128K BANKS AND
006430  006267  013314                  ASR     $HIOCT          ;NUMBER OF 8K BANK WITHIN
006434  006016                          ROR     (SP)            ;128K BANK
006436  006367  013306                  ASL     $HIOCT          ;MAKE NUMBER OF 128K BANKS INDEX
006442  012700  001714                  MOV     #BITPT,R0       ;GET START OF TEST TABLE
006446  066700  013276                  ADD     $HIOCT,R0       ;ADD INDEX
006452  012701  000012                  MOV     #12,R1          ;SET UP TO ALIGN 8K BANK COUNT
006456  006016                  1$:     ROR     (SP)            ;ALIGN COUNT
006460  077102                          SOB     R1,1$           ;LOOP UNTIL DONE
006462  052710  000001                  BIS     #BIT0,(R0)      ;INIT 8K BANK POINTER IN 128K WORD
006466  005716                  2$:     TST     (SP)            ;SHIFT BIT UNTIL COUNT = 0
006470  001403                          BEQ     3$              ;WHEN ZERO  DONE
006472  006310                          ASL     (R0)            ;SHIFT BIT
006474  005316                          DEC     (SP)            ;SUBTRACT FROM COUNT
006476  000773                          BR      2$              ;LOOP BACK
006500  012702  001514          3$:     MOV     #MEMMAP,R2      ;GET FIRST ADDRESS OF MEMORY TABLE
006504  066702  013240                  ADD     $HIOCT,R2       ;ADD INDEX
006510  031012                          BIT     (R0),(R2)       ;IS BIT JUST LOCATED,SET IN MEM. TABLE
006512  001725                          BEQ     MANUL1          ;IF NO BANK EXISTS...REASK
006514                          4$:
006514  004567  013232                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
006520  024540                          .WORD   LADMES          ;ADDRESS OF MESSAGE TO BE TYPED
```

```
                                                        ;"LAST ADDRESS:"
1863                            ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $RDOCT ROUTINE
                                ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
     006522  106746                    MFPS    (SP)            ;PUT THE PROCESSOR STATUS ON THE STACK
     006524  105066  000001            CLRB    1(SP)           ;HIGH BYTE CLEARED TO INSURE KERNEL MODE
                                                                ;ON PSW RETURN.
     006530  004767  013042            JSR     PC,     $RDOCT  ;GO TO THE SUBROUTINE
1864 006534  005716                    TST     (SP)            ;CHECK IF LOW BITS ZERO
1865 006536  001010                    BNE     5$              ;IF NOT,NO DEFAULT
1866 006540  005767  013204            TST     $HIOCT          ;CHECK IF HIGH BITS ZERO
1867 006544  001005                    BNE     5$              ;IF NOT,NO DEFAULT
1868 006546  016716  172402            MOV     $TMP2,(SP)      ;IF BOTH ZERO,FILL IN DEFAULT
1869 006552  016767  172400  013170    MOV     $TMP3,$HIOCT    ;FOR LOW AND HIGH BITS
1870 006560  020467  013164    5$:     CMP     R4,$HIOCT       ;CHECK FOR LAST ADDR. BELOW FIRST
1871 006564  101353                    BHI     4$              ;IF YES....REASK
1872 006566  103402                    BLO     6$              ;IF LAST HIGHER GO ON
1873 006570  020316                    CMP     R3,(SP)         ;IF EQUAL CHECK LOW BITS
1874 006572  101350                    BHI     4$              ;IF LOW BITS LOWER...REASK
1875 006574  012700  000020    6$:     MOV     #20,R0          ;TABLE COUNTER
1876 006600  012701  001714            MOV     #BITPT,R1
1877 006604  012702  001614            MOV     #SAVTST,R2
1878 006610  052122            7$:     BIS     (R1)+,(R2)+     ;STORE BITPT IN SAVTST MAP
1879 006612  077002                    SOB     R0,7$           ;DO 16. TIMES
1880 006614  020367  013130            CMP     R3,$HIOCT       ;COMPARE HIGH BITS OF FIRST TO LAST
1881 006620  103403                    BLO     8$              ;IF LOWER SEE IF NEXT BANK EXISTS
1882 006622  101027                    BHI     11$             ;IF HIGHER MUST BE LAST BANK
1883 006624  020416                    CMP     R4,(SP)         ;IF EQUAL CHECK LOW BITS
1884 006626  101025                    BHI     11$             ;IF LOW BITS FIRST HIGHER LAST BANK
1885 006630  062704  040000    8$:     ADD     #40000,R4       ;UPDATE TO NEXT BANK
1886
1887 006634  005503                    ADC     R3
1888 006636  012700  000017            MOV     #17,R0          ;SET UP TO UPDATE POINTER
1889 006642  012701  001714            MOV     #BITPT,R1       ;GET START ADDRESS OF TABLE
1890 006646  006321                    ASL     (R1)+           ;ROTATE POINTER WITHIN
1891 006650  006121            9$:     ROL     (R1)+           ;TABLE
1892 006652  077002                    SOB     R0,9$           ;UNTIL ALL LOCATIONS DONE
1893 006654  103423                    BCS     13$             ;BANK DOESN'T MAP
1894 006656  012700  000020            MOV     #20,R0          ;SET UP TO SEE IF ONE EXISTS
1895 006662  012701  001714            MOV     #BITPT,R1       ;GET START ADDRESS OF POINTER TABLE
1896 006666  012702  001514            MOV     #MEMMAP,R2      ;GET START ADDRESS OF MEMORY TABLE
1897 006672  032122            10$:    BIT     (R1)+,(R2)+     ;TEST IF BANK EXISTS
1898 006674  001337                    BNE     6$              ;IF MATCH UPDATE
1899 006676  077003                    SOB     R0,10$          ;DO ALL OF TABLE
1900 006700  000753                    BR      8$
1901 006702  012700  000020    11$:    MOV     #20,R0          ;MAKE SURE LAST BANK
1902 006706  012701  001714            MOV     #BITPT,R1       ;WAS MAPPED BY MEMORY
1903 006712  012702  001514            MOV     #MEMMAP,R2      ;SIZING ROUTINE
1904 006716  032122            12$:    BIT     (R1)+,(R2)+     ;
1905 006720  001007                    BNE     14$
1906 006722  077003                    SOB     R0,12$
1907 006724  012706  001100    13$:    MOV     #STACK,SP       ;RESET STACK
1908 006730  004567  013016            JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
     006734  024563                    .WORD   BADADR          ;ADDRESS OF MESSAGE TO BE TYPED
                                                                ;"?ADDRESS IN UNMAPPED BANK?"
1909 006736  000604                    BR      MANUAL          ;LOOP BACK AND START OVER
1910 006740  012706  001100    14$:    MOV     #STACK,SP       ;RESET STACK
1911 006744  004567  013002            JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
```

```
        006750  024620                          .WORD   CONST           ;ADDRESS OF MESSAGE TO BE TYPED
                                                                        ;"SELECT CONSTANT: "
 1912                                   ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $RDOCT ROUTINE
                                        ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
        006752  106746                          MFPS    -(SP)           ;PUT THE PROCESSOR STATUS ON THE STACK
        006754  105066  000001                  CLRB    1(SP)           ;HIGH BYTE CLEARED TO INSURE KERNEL MODE
                                                                        ;ON PSW RETURN.
        006760  004767  012612                  JSR     PC,     $RDOCT  ;GO TO THE SUBROUTINE
 1913   006764  012667  172774                  MOV     (SP)+,.CONST    ;SAVE THE DATA
 1914   006770  032777  000040  172132  MANUL2: BIT     #SW05,@SWR      ;IS VECTOR AREA PROTECTED?
 1915   006776  001403                          BEQ     START1          ;IF NO,GO ON
 1916   007000  012767  001000  172754          MOV     #1000,FSTADR    ;IF YES,SET STARTING ADDRESS = 1000
 1917
 1918
        .REPT   1
        ;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
        ;*  THE REST OF THE PROGRAM IS POSITION INDEPENDENT CODE, SO THAT IT CAN EXECUTE PROPERLY WHEN THE PROGRAM HAS BEEN RELOCATE
        ;*  THIS IS DONE SO THAT THE FIRST TWO BANKS OF MEMORY CAN BE EXERCISED IN EXACTLY THE SAME MANNER AS THE REST OF MEMORY.
        .REPT   1
        ;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*
 1923   007006  016706  172772          START1: MOV     .STACK, SP      ;SET STACK POINTER
 1924   007012  012700  000020                  MOV     #20,R0          ;SET UP TO LOAD 20 LOCATIONS
 1925   007016  016701  172766                  MOV     .SAVTST,R1      ;SOURCE TO BE SAVTST TABLE
 1926   007022  016702  172764                  MOV     .TSTMAP,R2      ;DESTINATIION TO BE TEST MAP
 1927   007026  012122                  1$:     MOV     (R1)+,(R2)+     ;INIT TESTING TABLE
 1928   007030  077002                          SOB     R0,1$           ;LOOP UNTIL DONE
 1929   007032  046767  171476  172514          BIC     PRGMAP,TSTMAP   ;DON'T TEST WHERE PROGRAM IS LOCATED.
 1930   007040  012767  007006  172030          MOV     #START1,$LPADR  ;INIT LOOP ADDRESS.
 1931   007046  066767  171460  172022          ADD     RELOCF, $LPADR
 1932   007054  004767  006660                  JSR     PC,     MAMF    ;SET UP MEMORY PARITY ERROR VECTOR
 1933   007060  005767  171452                  TST     MMAVA           ;CHECK FOR MEMORY MANAGEMENT AVAILABLE.
 1934   007064  001406                          BEQ     TST1            ;BRANCH IF NO MEM MGMT.
 1935   007066  032737  000001  177572          BIT     #BIT0,  @#SR0   ;CHECK IF MEM MGMT ENABLED.
 1936   007074  001002                          BNE     TST1            ;BR IF MEM MGMT ENABLED.
 1937   007076  004767  003676                  JSR     PC,     MMINIT  ;SET UP MEM MGMT REGISTERS.
 1938
 1939                                  .SBTTL SECTION 1:       MEMORY ADDRESS TESTS
 1948                                  ;;*****************************************************************
                                       ;*TEST 1        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.
                                       ;;*****************************************************************
        007102                         TST1:
        007102  004567  010424                  JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
 1949                                  ;*  UPWARDS WORD ADDRESSING.
 1950   007106  004467  004044                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
 1951   007112  010200                  1$:     MOV     R2,     R0      ;SET UP TO CALCULATE PHYSICAL ADDRESS
 1952   007114  004767  005156                  JSR     PC,     PHYADR  ;GET PHYSICAL ADDRESS INTO R0
 1953   007120  010012                  2$:     MOV     R0,     (R2)    ;WRITE VALUE OF ADDRESS INTO ADDRESS
 1954   007122  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
        007124  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
        007126  001405                          BEQ     30015$          ;BRANCH OVER ERROR CALL IF GOOD DATA.
        007130  004767  007646          30014$: JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
```

```
        007134  004767  010704                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
        007140  000002                          .WORD   2               ;ERROR TYPE CODE.
        007142                          30015$:
  1955  007142  062700  000002                  ADD     #2,     R0      ;ADD #2 TO PHYSICAL ADDRESS
  1956  007146  077514                          SOB     R5,     2$      ;BRANCH IF MORE IN CURRENT BLOCK.
        007150  004767  004214                  JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
  1957
  1958                                  ;*  CHECK THAT VALUE OF MEMORY ADDRESS WAS WRITTEN CORRECTLY
  1959                                  ;*  DOWNWARDS WORD ADDRESSING.
  1960  007154  004467  004074                  JSR     R4,     INITDN  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
  1961  007160  010200                  3$:     MOV     R2,     R0      ;SET UP TO CALCULATE PHYSICAL ADDRESS
  1962  007162  162700  000002                  SUB     #2,     R0      ;GET LAST ADDRESS OF BLOCK
  1963  007166  004767  005104                  JSR     PC,     PHYADR  ;GET PHYSICAL ADDRESS INTO R0
  1964  007172  014201                  4$:     MOV     -(R2),  R1      ;GET THE DATA FROM MEMORY
  1965  007174  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
        007176  001405                          BEQ     30017$          ;BRANCH OVER ERROR CALL IF GOOD DATA.
        007200  004767  007552          30016$: JSR     PC,     SPRNTO  ;SET UP VALUES FOR ERROR PRINTING.
        007204  004767  010634                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
        007210  000002                          .WORD   2               ;ERROR TYPE CODE.
        007212                          30017$:
  1966  007212  162700  000002                  SUB     #2,     R0      ;DEC DATA BY 2
  1967  007216  077513                          SOB     R5,     4$      ;BRANCH IF MORE IN CURRENT BLOCK.
        007220  004767  004510                  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.
                                ;;*****************************************************************
     007224                     TST2:
     007224  004567  010302             JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
1970                            ;*  UPWARDS BYTE ADDRESSING.
1971 007230  004467  003722             JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1972 007234  010200             1$:     MOV     R2,     R0      ;SET UP TO CALCULATE PHYSICAL ADDRESS
1973 007236  004767  005034             JSR     PC,     PHYADR  ;GET PHYSICAL ADDRESS INTO R0
1974 007242  006305                     ASL     R5              ;MAKE TEST COUNTER BYTE VALUE
1975 007244  110022             2$:     MOVB    R0,     (R2)+   ;WRITE VALUE OF ADDRESS INTO ADDRESS
1976 007246  005200                     INC     R0              ;ADD ONE TO PHYSICAL ADDRESS
1977 007250  077503                     SOB     R5,     2$      ;BRANCH IF MORE IN CURRENT BLOCK.
     007252  004767  004112             JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
1978
1979                            ;*  CHECK THAT VALUE OF MEMORY ADDRESS WAS WRITTEN CORRECTLY
1980                            ;*  DOWNWARDS BYTE ADDRESSING.
1981 007256  004467  003772             JSR     R4,     INITDN  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1982 007262  010200             3$:     MOV     R2,     R0      ;SET UP TO CALCULATE PHYSICAL ADDRESS
1983 007264  005300                     DEC     R0              ;GET LAST BYTE ADDRESS OF BLOCK
1984 007266  004767  005004             JSR     PC,     PHYADR  ;GET PHYSICAL ADDRESS INTO R0
1985 007272  006305                     ASL     R5              ;MAKE TEST COUNTER BYTE VALUE
1986 007274  114201             4$:     MOVB    -(R2),  R1      ;GET THE DATA FROM MEMORY
1987 007276  120001                     CMPB    R0,     R1      ;CHECK THE DATA...LO BYTE ONLY VALID.
1988 007300  001405                     BEQ     30019$          ;BRANCH OVER ERROR CALL IF GOOD DATA.
     007302  004767  007450     30018$: JSR     PC,     SPRNTO  ;SET UP VALUES FOR ERROR PRINTING.
     007306  004767  010532             JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
     007312  000003                     .WORD   3               ;ERROR TYPE CODE.
     007314                     30019$:
1989 007314  005300                     DEC     R0              ;DEC DATA BY 1
1990 007316  077512                     SOB     R5,     4$      ;BRANCH IF MORE IN CURRENT BLOCK.
     007320  004767  004410             JSR     PC,     MMDOWN  ;FIND NEXT BLOCK AND LOOP TO 3$.
1991
1992                            ;;*****************************************************************
                                ;*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.
                                ;;*****************************************************************
     007324                     TST3:
     007324  004567  010202             JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
1993                            ;*  DOWNWARDS WORD ADDRESSING.
1994 007330  004467  003720             JSR     R4,     INITDN  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1995 007334  010200             1$:     MOV     R2,     R0      ;SET UP TO CALCULATE PHYSICAL ADDRESS
1996 007336  162700  000002             SUB     #2,     R0      ;GET LAST ADDRESS OF BLOCK
1997 007342  004767  004730             JSR     PC,     PHYADR  ;GET PHYSICAL ADDRESS INTO R0
1998 007346  005100                     COM     R0              ;COMPLEMENT THE ADR
1999 007350  010042             2$:     MOV     R0,      (R2)   ;PUT DATA INTO MEMORY
```

```
                                        ADD     #2,     R0      ;+2 TO DATA--ADR GOES DOWN SO COM GOES UP
                                        SOB     R5,     2$      ;BRANCH IF MORE IN CURRENT BLOCK.
007360  004767  004350                  JSR     PC,     MMDOWN  ;FIND NEXT BLOCK AND LOOP TO 1$.

                                ;*  CHECK COMPLEMENT DATA WRITTEN DOWN
                                ;*  UPWARDS WORD ADDRESSING.
007364  004467  003566                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
007370  010200                  3$:     MOV     R2,     R0      ;SET UP TO CALCULATE PHYSICAL ADDRESS
007372  004767  004700                  JSR     PC,     PHYADR  ;GET PHYSICAL ADDRESS INTO R0
007376  005100                          COM     R0              ;COMPLEMENT IT
007400                          4$:
007400  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
007402  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
007404  001405                          BEQ     30021$          ;BRANCH OVER ERROR CALL IF GOOD DATA.
007406  004767  007370          30020$: JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
007412  004767  010426                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
007416  000002                          .WORD   2               ;ERROR TYPE CODE.
007420                          30021$:
007420  162700  000002                  SUB     #2,     R0      ;COUNT DOWN WITH ADDRESS
007424  077513                          SOB     R5,     4$      ;BRANCH IF MORE IN CURRENT BLOCK
007426  004767  003736                  JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 3$.

                                ;;*****************************************************************
                                ;*TEST 4        WRITE BANK # INTO ALL ADDRESSES IN A 8K 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.
                                ;;*****************************************************************
007432                          TST4:
007432  004567  010074                  JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
                                ;*  UPWARDS BYTE ADDRESSING.
007436  004467  003514                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
007442  004767  004726          1$:     JSR     PC,     BANKNO  ;GET THE BANK NUMBER INTO R0
007446  006305                          ASL     R5              ;MAKE TEST COUNTER BYTE VALUE
007450  110022                  2$:     MOVB    R0,     (R2)+   ;WRITE BANK # INTO ALL ADDRESSES
007452  077502                          SOB     R5,     2$      ;BRANCH IF MORE IN CURRENT BLOCK.
007454  004767  003710                  JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.

                                ;*  CHECK THAT DATA WRITTEN ABOVE CAN BE READ
                                ;*  UPWARDS BYTE ADDRESSING.
007460  004467  003472                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
007464  004767  004704          3$:     JSR     PC,     BANKNO  ;GET THE BANK NUMBER INTO R0
007470  006305                          ASL     R5              ;MAKE TEST COUNTER BYTE VALUE
007472  112201                  4$:     MOVB    (R2)+,  R1      ;READ THE DATA OUT OF MEMORY
007474  120001                          CMPB    R0,     R1      ;CHECK DATA....LOW BYTE ONLY VALID
007476  001405                          BEQ     30023$          ;BRANCH OVER ERROR CALL IF GOOD DATA.
007500  004767  007260          30022$: JSR     PC,     SPRNT1  ;SET UP VALUES FOR ERROR PRINTING.
007504  004767  010334                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
007510  000003                          .WORD   3               ;ERROR TYPE CODE.
007512                          30023$:
007512  077511                          SOB     R5,     4$      ;BRANCH IF MORE IN CURRENT BLOCK
007514  004767  003650                  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.
                                        ;;*************************************************************
     007520                             T5T5:
     007520  004567  010006                     JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
2032                                    ;*  DOWNWARDS BYTE ADDRESSING.
2033 007524  004467  003524                     JSR     R4,     INITDN  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2034 007530  004767  004640            1$:      JSR     PC,     BANKNO  ;GET THE BANK NUMBER INTO R0
2035 007534  006305                             ASL     R5              ;MAKE TEST COUNTER BYTE VALUE
2036 007536  005100                             COM     R0              ;1'S COMPLEMENT OF BANK #
2037 007540  110042                    2$:      MOVB    R0,     -(R2)   ;PUT 1'S COM OF BANK # INTO MEMORY
2038 007542  077502                             SOB     R5,     2$      ;BRANCH IF MORE IN CURRENT BLOCK.
     007544  004767  004164                     JSR     PC,     MMDOWN  ;FIND NEXT BLOCK AND LOOP TO 1$.
2039
2040                                    ;*  CHECK THAT DATA WRITTEN CAN BE READ.
2041                                    ;*  DOWNWARDS BYTE ADDRESSING.
2042 007550  004467  003500                     JSR     R4,     INITDN  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2043 007554  004767  004614            3$:      JSR     PC,     BANKNO  ;GET THE BANK # INTO R0
2044 007560  006305                             ASL     R5              ;MAKE TEST COUNTER BYTE VALUE
2045 007562  005100                             COM     R0              ;SET 1'S COMPLEMENT OF BANK #
2046 007564  114201                    4$:      MOVB    -(R2),  R1      ;READ DATA OUT OF MEMORY
2047 007566  120001                             CMPB    R0,     R1      ;CHECK DATA...LOW BYTE ONLY VALID
2048 007570  001405                             BEQ     30025$          ;BRANCH OVER ERROR CALL IF GOOD DATA.
     007572  004767  007160            30024$: JSR     PC,     SPRNTO  ;SET UP VALUES FOR ERROR PRINTING.
     007576  004767  010242                     JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
     007602  000003                             .WORD   3               ;ERROR TYPE CODE.
     007604                             30025$:
2049 007604  077511                             SOB     R5,     4$      ;BRANCH IF MORE IN CURRENT BLOCK.
     007606  004767  004122                     JSR     PC,     MMDOWN  ;FIND NEXT BLOCK AND LOOP TO 3$.
2050
2051
2052                                    ;*  IF PROGRAM HAS RELOCATED TO UPPER BOUNDARY,THE ADDRESSING
2053                                    ;*  TESTS WILL BE EXECUTED FOR ALL BANKS AND WORST CASE NOISE TESTING
2054                                    ;*  WILL BE LIMITED TO BANKS 0,1.
2055                                    ;*  ALL OTHER BANKS WILL BE EXERCISED FOR WORST CASE NOISE TESTING
2056                                    ;*  WHEN THE PROGRAM OCCUPIES BANKS 0,1.
2057
2058 007612  032767  000001  170714    SECT2:  BIT     #BIT0,PRGMAP    ;IS PROGRAM RELOCATED
2059 007620  001014                             BNE     2$              ;IF NO,CONTINUE TESTING ALL BANKS
2060 007622  012700  000020                     MOV     #20,R0          ;CLEAR ALL LOCATIONS OF TEST MAP
2061 007626  016701  172160                     MOV     .TSTMAP,R1      ;TEST MAP TABLE
2062 007632  005021                    1$:      CLR     (R1)+
2063 007634  077002                             SOB     R0,1$           ;LOOP UNTIL DONE
2064 007636  016700  171752                     MOV     SAVTST,R0       ;GET LOWER 128K MAP
2065 007642  042700  177776                     BIC     #177776,R0      ;DO ONLY FIRST BANK
2066 007646  010067  171702                     MOV     R0,TSTMAP       ;FOR DATA TESTS
2067 007652                             2$:                             ;CONTINUE TESTING
2068
2069
2070                                    .SBTTL SECTION 2:       WORST CASE NOISE TESTS
2071                                    ;;*************************************************************
2072                                    ;*  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.
                                ;;*****************************************************************
     007652                     TST6:
     007652  004567  007654             JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
2076 007656  016700  172102     TST6A:  MOV     .CONST, R0      ;GET USER CONSTANT
2077 007662  004467  003270             JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2078 007666  010022             1$:     MOV     R0,     (R2)+   ;WRITE CONSTANT INTO MEMORY.
2079 007670  077502                     SOB     R5,     1$      ;BRANCH IF MORE IN CURRENT BLOCK
     007672  004767  003472             JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
2080
2084                            ;;*****************************************************************
                                ;*TEST 7         READ MEMORY AND COMPARE TO CONSTANT.
                                ;*  IMPORTANT:  THIS TEST SHOULD NOT BE RUN WITHOUT FIRST RUNNING TEST $TN.
                                ;;*****************************************************************
     007676                     TST7:
     007676  004567  007630             JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
2085 007702  016700  172056             MOV     .CONST, R0      ;GET USER CONSTANT
2086 007706  004467  003244             JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2087 007712                     1$:
     007712  012201                     MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
     007714  020001                     CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
     007716  001405                     BEQ     30027$          ;BRANCH OVER ERROR CALL IF GOOD DATA.
     007720  004767  007056     30026$: JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
     007724  004767  010114             JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
     007730  000004                     .WORD   4               ;ERROR TYPE CODE.
     007732                     30027$:
2088 007732  077511                     SOB     R5,     1$      ;BRANCH IF MORE IN CURRENT BLOCK
     007734  004767  003430             JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
2090                            ;*  SPECIAL CHECK TO SEE IF TEST 6 IS SELECTED THRU THE SWR.
2091                            ;*  ALLOWS THE OPERATOR TO SWITCH BACK AND FORTH BETWEEN TESTS 6 AND 7
2092                            ;*  BY SIMPLY "TOGGLING" SW00 WHEN SW01, SW02, AND SW08 ARE SET.
2093 007740  032777  000400  171162         BIT     #SW08,  @SWR    ;CHECK THAT LOOP ON TEST BIT SET
2094 007746  001416                     BEQ     TST10           ;BRANCH IF NOT LOOP ON TEST
2095 007750  017746  171154             MOV     @SWR,   -(SP)   ;GET SWITCH REGISTER DATA.
2096 007754  042716  177740             BIC     #177740,(SP)    ;CLEAR NON-TEST-NUMBER SWITCHS.
2097 007760  022726  000006             CMP     #6,     (SP)+   ;CHECK IF TEST 6 IN SWITCHES.
2098 007764  001007                     BNE     TST10           ;BRANCH IF NOT TEST 6
2099 007766  162767  000001  171076     SUB     #1,     $TSTNM  ;RESET TEST NUM
2100 007774  162767  000024  171074     SUB     #TST7-TST6,$LPADR ;RESET LOOP ADR
2101 010002  000725                     BR      TST6A           ;GO TO TEST 6
2103
2104
                                ;;*****************************************************************
                                ;*TEST 10        WORSE CASE NOISE (PARITY) WORD TESTING
                                        ;*  CHECK MEMORY WITH A SERIES OF PATTERNS
                                ;;*****************************************************************
     010004                     TST10:
     010004  004567  007522             JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
```

N4


```
 2105 010010  016704  172014            MOV     .MPPAT, R4      ;INITIALIZE PATTERN TABLE POINTER
 2106 010014  004767  006020     1$:    JSR     PC,     CKPMER  ;CHECK FOR NON-TRAP PARITY MEMORY ERPORS.
 2107 010020  012400                    MOV     (R4)+,  R0      ;GET THE DATA PATTERN.
 2108 010022  001417                    BEQ     TST11           ;;BR IF END OF TABLE.
 2109 010024  004467  003126            JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
 2110 010030  010012               2$:  MOV     R0,     (R2)    ;PUT DATA PATTERN INTO MEMORY.
 2111 010032  012201                    MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
      010034  020001                    CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
      010036  001405                    BEQ     30029$          ;BRANCH OVER ERROR CALL IF GOOD DATA.
      010040  004767  006736  30028$.  JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
      010044  004767  007774            JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
      010050  000004                    .WORD                   ;ERROR TYPE CODE.
      010052                    30029$:
 2112 010052  077512                    SOB     R5,     2$      ;BRANCH IF MORE IN CURRENT BLOCK
      010054  004767  003310            JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 2$.
 2113 010060  000755                    BR      1$              ;BR BACK TO DO NEXT PATTERN
```

```
2115                                    ;;*********************************************************************
                                        ;*TEST 11       ROTATE A "0' BIT THROUGH A FIELD OF ONES.
                                        ;;*********************************************************************
     010062                             1$T11:
     010062  004567  007444                    JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
2116 010066  012700  177777                    MOV     #-1,    R0      ;SET CHECK WORD
2117 010072  004767  004364                    JSR     PC,     SETCON  ;PUT THE CONTENTS OF R0 IN ALL MEMORY.
2118 010076  004467  003054                    JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2119 010102  000241                    1$:     CLC                     ;CLEAR CARRY BIT IN PSW
2120 010104  004767  004370                    JSR     PC,     ROTATE
2121 010110  016201  177776                    MOV     -2(R2), R1      ;GET RESULT
2122 010114  103402                            BCS     63$             ;BRANCH IF 'C' BIT WAS SET
2123 010116  020001                            CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
     010120  001405                            BEQ     30030$          ;BRANCH OVER ERROR CALL IF GOOD DATA.
     010122  004767  006654            63$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
     010126  004767  007712                    JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
     010132  000005                            .WORD   5               ;ERROR TYPE CODE.
     010134                             30030$:
2124 010134  077516                            SOB     R5,     1$      ;BRANCH IF MORE IN CURRENT BLOCK
     010136  004767  003226                    JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
2125
2126                                    ;;*********************************************************************
                                        ;*TEST 12       ROTATE A "1" BIT THROUGH A FIELD OF ZEROS
                                        ;;*********************************************************************
     010142                             1$T12:
     010142  004567  007364                    JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
2127 010146  005000                            CLR     R0              ;SET CHECK WORD
2128 010150  004767  004306                    JSR     PC,     SETCON  ;PUT THE CONTENTS OF R0 IN ALL MEMORY
2129 010154  004467  002776                    JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2130 010160  000261                    1$:     SEC                     ;SET 'C' BIT IN PSW
2131 010162  004767  004312                    JSR     PC,     ROTATE  ;GO ROTATE '1' BIT
2132 010166  016201  177776                    MOV     -2(R2), R1      ;GET RESULT
2133 010172  103002                            BCC     63$             ;BRANCH IF 'C' IS CLEAR
2134 010174  020001                            CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
     010176  001405                            BEQ     30031$          ;BRANCH OVER ERROR CALL IF GOOD DATA.
     010200  004767  006576            63$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
     010204  004767  007634                    JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
     010210  000005                            .WORD   5               ;ERROR TYPE CODE.
     010212                             30031$:
2135 010212  077516                            SOB     R5,     1$      ;BRANCH IF MORE IN CURRENT BLOCK
     010214  004767  003150                    JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
2136
```

```
                                ;;*****************************************************************
                                ;*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.
                                ;;*****************************************************************
     010220                     TST13:
     010220  004567  007306             JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
2148 010224  005767  173142     WWPB0:  TST     MPRX            ;CHECK FOR ANY PARITY MEMORY.
2149 010230  001404                     BEQ     1$              ;BR IF NO PARITY MEMORY.
2150 010232  032777  000100  170670     BIT     #SW06,  @SWR    ;CHECK FORINHIBIT PARITY SWITCH.
2151 010240  001402                     BEQ     2$              ;BR IF NOT SET.
2152 010242  000167  000760     1$:     JMP     TST14           ;SKIP THIS TEST IF NO PARITY MEMORY PRESENT.
2153 010246  005000             2$:     CLR     R0              ;ZERO TO BE PUT IN ALL MEMORY.
2154 010250  004767  004206             JSR     PC,     SETCON  ;ROUTINE TO LOAD ALL MEMORY.
2155 010254  004467  002676             JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2156
2157                                                             ;RETURN FOR MMUP
2158 010260  006305                     ASL     R5              ;MAKE TEST COUNTER BYTE VALUE
2159 010262  012700  000020     WWPBYT: MOV     #20,    R0      ;TABLE COUNT
2160 010266  016701  171524             MOV     .BITPT, R1      ;BANK POINTER TABLE
2161 010272  016703  171516             MOV     .PMEMAP,R3      ;PARITY TABLE
2162 010276  032123             1$:     BIT     (R1)+,(R3)+     ;CHECK IF CURRENT BANK HAS PARITY MEMORY
2163 010300  001004                     BNE     2$              ;BRANCH IF PARITY MEMORY
2164 010302  077003                     SOB     R0,     1$      ;CHECK TILL END OF TABLE
2165 010304  060502                     ADD     R5,     R2      ;IF NONE FOUND POINT R2 TO
2166                                                             ;FIRST ADDR OF NEXT BANK.
2167 010306  000167  000704             JMP     WWPB6           ;FIND NEXT BLOCK
2168 010312  004767  005466     2$:     JSR     PC,     SETAE   ;SET ACTION ENABLE (EVEN IF BANK0.)
2169 010316  004767  005516             JSR     PC,     CKPMER  ;CHECK FOR ANY NON TRAP PARITY ERRORS.
2170 010322  020227  000114     WWPB1:  CMP     R2,     #114    ;CHECK IF POINTING TO PARITY ERROR VECTOR.
2171 010326  001004                     BNE     3$              ;BR IF NOT AT VECTOR.
2172 010330  062702  000004             ADD     #4,     R2      ;SKIP PARITY VECTOR.
2173 010334  162705  000004             SUB     #4,     R5      ;SKIP PARITY VECTOR
2174                                                             ;SUBTACT 4 FROM TEST COUNTER
2175 010340  111201             3$:     MOVB    (R2),   R1      ;CHECK IF BYTE STILL CLEARED.
2176 010342  001405                     BEQ     30033$          ;BRANCH OVER ERROR CALL IF GOOD DATA.
     010344  004767  006344     30032$: JSR     PC,     SPRNT   ;SET UP VALUES FOR ERROR PRINTING.
     010350  004767  007470             JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
     010354  000011                     .WORD   11              ;ERROR TYPE CODE.
     010356                     30033$:
2177 010356  105067  171376             CLRB    OEFLG           ;CLEAR ODD/EVEN FLAG.
2178 010362  112700  000252             MOVB    #252,   R0      ;SET UP DATA...EVEN, SETS PARITY BIT.
2179 010366  110012             WWPB2:  MOVB    R0,     (R2)    ;MOV DATA INTO TEST LOCATION.
2180 010370  016703  171430             MOV     .MPRX,  R3      ;GET PARITY REGISTER TABLE POINTER.
2181 010374  056773  171366  000000 10$:     BIS     WWP,@(R3)       ;SET WRITE WRONG PARITY.
2182 010402  052733  000001             BIS     #AE,@(R3)+
2183 010406  005713                     TST     (R3)            ;CHECK FOR TABLE TERMINATOR.
2184 010410  001371                     BNE     10$             ;BR IF MORE REGS IN TABLE.
2185                                ;*  SET WRONG PARITY IN LOCATION UNDER TEST.
2186 010412  110012                     MOVB    R0,     (R2)    ;WRITE SAME DATA (EXCEPT PARITY) VIA DATOB.
2187 010414  016703  171404             MOV     .MPRX,  R3      ;GET PARITY REG TABLE POINTER.
2188 010420  046733  171342     11$:    BIC     WWP,    @(R3)+  ;CLEAR WRITE WRONG PARITY.
```

```
 2189 010424  005713                          TST     (R3)            ;CHECK FOR TABLE TERMINATOR.
 2190 010426  001374                          BNE     11$             ;BR IF MORE PARITY REGISTERS.
 2191 010430  005767  171342                  TST     LSIFLG          ;CHECK IF RUNNING ON EARLIER LSI-11/2
 2192 010434  001402                          BEQ     12$             ;BR IF 11/23 OR LATER PROCESSOR
 2193 010436  105712                          TSTB    (R2)            ;PARITY TRAP LOGIC NOT AVAILABLE ON LSI11/2
 2194 010440  000433                          BR      PBTRP1          ;SET PARITY ERROR WITH READ AND BYPASS
 2195                                                                 ;PARITY TRAP AND TEST PARITY CSR.
 2196 010442  016737  171360  000114  12$:    MOV     .PBTRP, @#PARVEC ;SET UP VECTOR FOR EXPECTED TRAP.
 2197                                ;*  DETECT WRONG PARITY VIA DATIO; DATOB SHOULDN'T EXECUTE.
 2198                                ;;VK*************************************************************************
 2199                                ;;VK CHECK FOR KDJ11-DA (11/53) PROCESSOR
 2200                                ;;VK*************************************************************************
 2201 010450  005767  171324                  TST     KDJDA
 2202 010454  001404                          BEQ     14$
 2203 010456  105712                          TSTB    (R2)
 2204 010460  005767  173260                  TST     START
 2205 010464  000401                          BR      16$
 2206                                ;;VK*************************END OF KDJDA CHECK*************************
 2207 010466  105412                  14$:    NEGB    (R2)            ;DATIO (DATOB AND COM PARITY BIT.)
 2208                                ;*  SHOULD HAVE TRAPPED TO PBTRP.
 2209 010470  016737  171336  000114  16$:    MOV     .PESRV, @#PARVEC ;RESET VECTOR FOR UNEXPECTED TRAPS.
 2210 010476  004767  006254          30034$: JSR     PC,     SPRNTO  ;SET UP VALUES FOR ERROR PRINTING.
      010502  004767  007336                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
      010506  000012                          .WORD   12              ;ERROR TYPE CODE.
 2211 010510  004767  006156                  JSR     PC,CLRPAR       ;CLEAR ALL PARITY CSRS BEFORE NEXT BYTE
 2212 010514  000167  000442                  JMP     WWPB4           ;SKIP TRAP SERVICE.
 2213
 2214
 2215
 2216                                ;*  EXPECTED PARITY MEMORY TRAPS COME HERE.
 2217 010520  016737  171306  000114  PBTRP:  MOV     .PESRV, @#PARVEC ;RESET PARITY VECTOR FOR UNEXPECTED TRAPS.
 2218 010526  022626                          CMP     (SP)+,  (SP)+   ;RESET THE STACK POINTER AFTER TRAP.
 2219 010530  016703  171266          PBTRP1: MOV     .MPR0,  R3      ;GET PARITY REG AND MAP TABLE POINTER.
 2220 010534  032713  000001          21$:    BIT     #BIT0,  (R3)    ;CHECK IF THIS REGISTER EXISTS.
 2221 010540  001003                          BNE     22$             ;BR IF IT DOESN'T EXIST.
 2222 010542  017301  000000                  MOV     @(R3),  R1      ;GET THE CONTENTS.
 2223 010546  100413                          BMI     23$             ;BR IF ERROR FLAG SET.
 2224 010550  062703  000044          22$:    ADD     #44,    R3      ;MOVE POINTER TO NEXT REG.
 2225 010554  020367  171244                  CMP     R3,     .MPRX   ;CHECK FOR END OF TABLE.
 2226 010560  103765                          BLO     21$             ;BR IF MORE REGISTERS.
 2227 010562  004767  006170          30035$: JSR     PC,     SPRNTO  ;SET UP VALUES FOR ERROR PRINTING.
      010566  004767  007252                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
      010572  000013                          .WORD   13              ;ERROR TYPE CODE.
 2228 010574  000572                          BR      WWPB4           ;EXIT AFTER ERROR.
 2229 010576  004567  005344          23$:    JSR     R5,     PARMAT  ;CHECK FOR MAP OF THIS REGISTER
 2230 010602  000405                          BR      24$             ;BRANCH IF THIS PARITY REGISTER
 2231                                                                 ;CONTROLS THIS BANK.
 2232                                                                 ;RETURN +2,ERROR ,CANNOT FIND MEM BANK
 2233                                                                 ;IN PARITY MAP TABLE.
 2234 010604  004767  006142          30036$: JSR     PC,     SPRNTP  ;SET UP VALUES FOR ERROR PRINTING.
      010610  004767  007230                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
      010614  000014                          .WORD   14              ;ERROR TYPE CODE.
 2235 010616                          24$:
      010616  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
 2236 010620  010200                          MOV     R2,     R0      ;GET THE ADDRESS POINTER.
 2237 010622  042700  003777                  BIC     #3777,  R0      ;CLEAR LOW ADDRESS BITS.
 2238 010626  000300                          SWAB    R0              ;SHIFT 6 PLACES RIGHT.
```

```
 2239 010630  006300                        ASL     R0
 2240 010632  006300                        ASL     R0
 2241 010634  005767  167676                TST     MMAVA           ;CHECK FOR MEM MGMT.
 2242 010640  001411                        BEQ     25$             ;BR IF NO MEM MGMT.
 2243 010642  042700  177600                BIC     #177600,R0      ;CLEAR BANK BITS
 2244 010646  063700  172344                ADD     @#KIPAR2,R0     ;ADD MEM MGMT OFFSET.
 2245 010652  032702  020000                BIT     #BIT13, R2      ;CHECK FOR PAR3 REF. ADDRESSING
 2246 010656  001402                        BEQ     25$             ;BR IF R2 IS ADDRESSING PAR2
 2247 010660  062700  000200                ADD     #200,R0         ;SET UP EXPECTED WITH PAR3
 2248 010664  052700  100001        25$:    BIS     #BIT15+BIT0,R0  ;SET ERROR AND AE BIT IN CHECK WORD.
 2249 010670  016367  000042  170610        MOV     42(R3), RESRVD  ;GET APPROPIATE MASK.
 2250 010676  046700  170604                BIC     RESRVD, R0      ;CLEAR PARITY REG BITS RESERVED FOR FUTURE.
 2251 010702  046701  170600                BIC     RESRVD, R1      ;CLEAR PARITY REG BITS RESERVED FOR FUTURE.
 2252                             ;NOTE: THE ABOVE INSTRUCTION (2 WORDS) CAN BE NOP'ED FOR UNMIXED MEMORY TYPES.
 2253 010706  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
      010710  001405                        BEQ     30038$          ;BRANCH OVER ERROR CALL IF GOOD DATA.
      010712  004767  006034        30037$: JSR     PC,     SPRNTP  ;SET UP VALUES FOR ERROR PRINTING.
      010716  004767  007122                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
      010722  000015                        .WORD   15              ;ERROR TYPE CODE.
      010724                        30038$:
 2254 010724  005767  167606                TST     MMAVA           ;CHECK FOR 22 BIT ADDRESSING
 2255 010730  100034                        BPL     100$            ;BR IF 22 BIT ADDR NOT ENABLED
 2256 010732  022737  010000  172344        CMP     #10000,@#KIPAR2 ;CHECK IF ABOVE 128KW
 2257 010740  101030                        BHI     100$            ;BR IF KIPAR2 LESS THAN 128KW
 2258 010742  010046                        MOV     R0,-(SP)        ;;PUSH R0 ON STACK
 2259 010744  052773  040000  000000        BIS     #BIT14,@(R3)    ;SET BIT14 TO ENABLE BITS 18 21
 2260                                                                ;INTO BITS 8-5 OF PARITY CSR
 2261 010752  013700  172344                MOV     @#KIPAR2,R0     ;SETUP EXPECTED FOR PARITY CSR
 2262 010756  042700  007777                BIC     #7777,R0        ;BITS 18-21 STORED IN
 2263 010762  000300                        SWAB    R0              ;BITS 5-8.
 2264 010764  006300                        ASL     R0              ;A18-21 IN BITS 5-8
 2265 010766  052700  140001                BIS     #BIT15!BIT14!AE,R0;SAVE EXPECTED PARITY CSR
 2266 010772  017301  000000                MOV     @(R3),R1        ;GET CONTENTS OF PARITY CSR
 2267 010776  042701  030032                BIC     #30032,R1       ;CLEAR MASKED BITS
 2268 011002  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
      011004  001405                        BEQ     30040$          ;BRANCH OVER ERROR CALL IF GOOD DATA.
      011006  004767  005740        30039$: JSR     PC,     SPRNTP  ;SET UP VALUES FOR ERROR PRINTING.
      011012  004767  007026                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
      011016  000015                        .WORD   15              ;ERROR TYPE CODE.
      011020                        30040$:
 2269 011020  012600                        MOV     (SP)+,R0        ;;POP STACK INTO R0
 2270 011022  005073  000000        100$:   CLR     @(R3)           ;CLEAR REG INCLUDING ACTION ENABLE.
 2271 011026  010346                        MOV     R3,-(SP)        ;;PUSH R3 ON STACK
 2272 011030  062703  000044        26$:    ADD     #44,    R3      ;UPDATE POINTER TO NEXT PARITY REG + MAP.
 2273 011034  020367  170764                CMP     R3,     .MPRX   ;CHECK FOR END OF TABLE.
 2274 011040  101014                        BHI     WWPB3           ;BR IF END OF TABLE REACHED.
 2275 011042  032713  000001                BIT     #BIT0,  (R3)    ;CHECK IF NEXT REG EXISTS.
 2276 011046  001370                        BNE     26$             ;BR IF THIS PARITY REG DOESN'T EXIST.
 2277 011050  017301  000000                MOV     @(R3),  R1      ;SAVE AND CHECK FOR ERROR FLAG.
 2278 011054  100365                        BPL     26$             ;BR IF NO ERROR FLAG.
 2279 011056  004767  005670        30041$: JSR     PC,     SPRNTP  ;SET UP VALUES FOR ERROR PRINTING.
      011062  004767  006756                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
      011066  000016                        .WORD   16              ;ERROR TYPE CODE.
 2280 011070  000757                        BR      26$             ;BR AFTER ERROR.
 2281 011072  111204                WWPB3:  MOVB    (R2),   R4      ;GET THE DATA FOR CHECKING.
 2282                               ;* READING THE DATA VIA DATI TO CHECK IT SHOULD CAUSE PARITY ERROR, BUT
 2283                               ;*  ACTION ENABLE IS NOT SET IN CONTROLLING REG, SO NO TRAP SHOULD OCCUR.
```

```
 2284 011074  111212                          MOVB    (R2),   (R2)    ;RESTORE RIGHT PARITY
 2285                                ;NOTE:    THE ABOVE INSTRUCTION CAN BE NOP'ED FOR PROCESSORS
 2286                                ;         WHICH DO ONLY DATOB TO DESTINATION OF MOVB INSTRUCTIONS.
 2287 011076  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
 2288 011100  017301  000000                  MOV     @(R3),  R1      ;READ THE PARITY REGISTER TO CHECK IT AGAIN.
 2289 011104  046701  170376                  BIC     RESRVD, R1      ;CLEAR PARITY REG BITS RESERVED FOR FUTURE.
 2290                                ;NOTE: THE ABOVE INSTRUCTION (2 WORDS) CAN BE NOP'ED FOR UNMIXED MEMORY TYPES.
 2291 011110  042700  000001                  BIC     #AE,    R0      ;CLEAR THE ACTION ENABLE BIT IN TEST DATA.
 2292 011114  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
      011116  001405                          BEQ     30043$          ;BRANCH OVER ERROR CALL IF GOOD DATA.
      011120  004767  005626          30042$: JSR     PC,     SPRNTP  ;SET UP VALUES FOR ERROR PRINTING.
      011124  004767  006714                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
      011130  000015                          .WORD   15              ;ERROR TYPE CODE.
      011132                          30043$:
 2293 011132  012773  000001  000000          MOV     #1,     @(R3)   ;CLEAR ALL BUT ACTION ENABLE.
 2294 011140  010401                          MOV     R4,     R1      ;GET DATA READ FROM MEMORY FOR TESTING.
 2295 011142  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
 2296 011144  120001                          CMPB    R0,     R1      ;CHECK THE DATA.
 2297 011146  001405                          BEQ     30045$          ;BRANCH OVER ERROR CALL IF GOOD DATA.
      011150  004767  005602          30044$: JSR     PC,     SPRNT0  ;SET UP VALUES FOR ERROR PRINTING.
      011154  004767  006664                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
      011160  000017                          .WORD   17              ;ERROR TYPE CODE.
      011162                          30045$:
 2298 011162  110012                  WWPB4:  MOVB    R0,     (R2)    ;RESTORE DATA.
 2299 011164  105712                          TSTB    (R2)            ;DO A DATI TO BE SURE RIGHT PARITY.
 2300 011166  012700  000253                  MOV     #253,   R0      ;SET ODD PARITY DATA.
 2301 011172  105167  170562                  COMB    OEFLG           ;CHECK IF DONE BOTH ODD AND EVEN PARITY.
 2302 011176  100002                          BPL     27$             ;BR IF DONE BOTH EVEN AND ODD.
 2303 011200  000167  177162                  JMP     WWPB2           ;LOOP BACK AND DO ODD(PARITY BIT CLR).
 2304 011204  005202                  27$:    INC     R2              ;MOVE POINTER TO NEXT MEMORY BYTE.
 2305 011206  005305                  WWPB5:  DEC     R5              ;CHECK FOR END OF BLOCK
  306 011210  001402                          BEQ     WWPB6           ;BR IF END OF BLOCK FOUND.
 2307 011212  000167  177104                  JMP     WWPB1           ;LOOP BACK TO TEST NEXT BYTE.
 2308 011216  004767  002146          WWPB6:  JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO WWPBYT
 2309 011222  004767  004512                  JSR     PC,     MAMF    ;GO RESET PARITY REGISTERS.
```

```
 2311                                   ;;*****************************************************************
                                        ;*TEST 14       RANDOM DATA  TESTING THRU PROGRAM CODE RELOCATION.
                                        ;;*****************************************************************
                                        TST14:
      011226                                    JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
      011226  004567  006300            RANTST: MOV     PC,     R3      ;GET CURRENT PROGRAM COUNTER.
 2312 011232  010703                            BIC     #7777,  R3      ;POINT TO BEGINNING OF CURRENT 2K BLOCK.
 2313 011234  042703  007777                    JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
 2314 011240  004467  001712            1$:     MOV     R2,     (SP)    ;SAVE MEMORY POINTER.
 2315 011244  010246                            MOV     R3,     -(SP)   ;SAVE "DATA" POINTER.
 2316 011246  010346                            MOV     R5,     TEMP1   ;SAVE ADDRESS COUNTER
 2317 011250  010567  170516            2$:     MOV     (R3)+,  (R2)+   ;MOV CODE INTO TEST MEMORY.
 2318 011254  012322                            BIT     #7777,  R3      ;CHECK FOR END OF "DATA TABLE'
 2319 011256  032703  007777                    BNE     3$              ;BRANCH IF MORE
 2320 011262  001002                            SUB     #10000, R3      ;RESET POINTER TO START OF "RANDOM DATA'
 2321 011264  162703  010000            3$:     SOB     R5,     2$      ;IF NOT END OF BLOCK,BRANCH TO 2$
 2322 011270  077507                            MOV     (SP)+,  R3      ;RESET "DATA" POINTER.
 2323 011272  012603                            MOV     (SP)+,  R2      ;RESET MEMORY POINTER.
 2324 011274  012602                            MOV     TEMP1,  R5      ;RESET ADDRESS COUNTER
 2325 011276  016705  170470            4$:     MOV     (R3)+,  R0      ;GET S/B DATA.
 2326 011302  012300                            MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
 2327 011304  012201                            CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
      011306  020001                            BEQ     30047$          ;BRANCH OVER ERROR CALL IF GOOD DATA.
      011310  001405
      011312  004767  005464          30046$: JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
      011316  004767  006522                    JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
      011322  000020                            .WORD   20              ;ERROR TYPE CODE.
      011324                            30047$:
 2328 011324  032703  007777                    BIT     #7777,  R3      ;CHECK FOR END OF "DATA TABLE"
 2329 011330  001002                            BNE     5$              ;BR IF MORE.
 2330 011332  162703  010000                    SUB     #10000, R3      ;RESET POINTER TO TOP OF "DATA TABLE".
 2331 011336                            5$:
      011336  077517                            SOB     R5,     4$      ;BRANCH IF MORE IN CURRENT BLOCK
      011340  004767  002024                    JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
 2332
 2333                                   .SBTTL SECTION 3:       INSTRUCTION EXECUTION TESTS.
 2358                                   ;;*****************************************************************
                                        ;*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.
                                        ;*  R3 - INSTRUCTION UNDER TEST (IUT).
```

```
                                        ;*  R4 = RTS R5 (CODE 205).
                                        ;*  R5 = BLOCK BOUNDARY BIT MASK.
                                        ;;*****************************************************************
        011344                          TST15:
        011344  004567  006162                  JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
2359    011350  012703  010412          DIDO:  MOV     #010412,R3      ;GET 'MOV R4,(R2)' INSTRUCTION (IUT).
2360    011354  012704  000205                 MOV     #205,   R4      ;GET 'RTS R5'
2361    011360  010400                          MOV     R4,     R0      ;SET UP S/B DATA AFTER EXECUTION.
2362    011362  004467  001570                 JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2363    011366  005305                  1$:    DEC     R5              ;DO ALL ADDRESSES -1
2364    011370  010322                          MOV     R3,     (R2)+   ;PUT IUT INTO FIRST LOC OF BLOCK.
2365    011372  010412                  2$:    MOV     R4,     (R2)    ;PUT 'RTS R5' FOLLOWING IUT.
2366    011374  004542                          JSR     R5,     -(R2)   ;GO EXECUTE THE IUT.
2367    011376  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM THE MEM ADR UNDER TEST.
2368    011400  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
        011402  001405                          BEQ     30049$          ;BRANCH OVER ERROR CALL IF GOOD DATA.
        011404  004767  005366  30048$: JSR     PC,     SPRNT3  ;SET UP VALUES FOR ERROR PRINTING.
        011410  004767  006430                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
        011414  000021                          .WORD   21              ;ERROR TYPE CODE.
        011416                          30049$:
2369    011416  010322                          MOV     R3,     (R2)+   ;PUT THE IUT INTO THE NEXT LOCATION.
2370    011420  077514                          SOB     R5,     2$      ;BRANCH IF MORE IN CURRENT BLOCK.
        011422  004767  001742                  JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
```

```
 2396                           ;;*****************************************************************
                                ;*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.
                                ;*****************************************************************
      011426                    TST16:
      011426  004567  006100            JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
 2397 011432  012703  110412    DIDBL:  MOV     #110412,R3      ;GET 'MOVB R4,(R2)' INSTRUCTION (IUT).
 2398 011436  012704  000205            MOV     #205,   R4      ;GET 'RTS R5'
 2399 011442  012700  110605            MOV     #110605,R0      ;SET UP S/B DATA AFTER EXECUTION.
 2400 011446  004467  001504            JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
 2401 011452  005305            1$:     DEC     R5              ;DO ALL ADDRESSES -1
 2402 011454  010322                    MOV     R3,     (R2)+   ;PUT IUT INTO FIRST LOC OF BLOCK.
 2403 011456  010412            2$:     MOV     R4,     (R2)    ;PUT 'RTS R5' FOLLOWING IUT.
 2404 011460  004542                    JSR     R5,     -(R2)   ;GO EXECUTE THE IUT.
 2405 011462  012201                    MOV     (R2)+,  R1      ;GET THE DATA FROM THE MEM ADR UNDER TEST.
 2406 011464  020001                    CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
      011466  001405                    BEQ     30051$          ;BRANCH OVER ERROR CALL IF GOOD DATA.
      011470  004767  005302    30050$: JSR     PC,     SPRNT3  ;SET UP VALUES FOR ERROR PRINTING.
      011474  004767  006344            JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
      011500  000021                    .WORD   21              ;ERROR TYPE CODE.
      011502                    30051$:
 2407 011502  010322                    MOV     R3,     (R2)+   ;PUT THE IUT INTO THE NEXT LOCATION.
 2408 011504  077504                    SOB     R5,     2$      ;BRANCH IF MORE IN CURRENT BLOCK.
      011506  004767  001656            JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
```

```
 2434                                ;;*****************************************************************
                                     ;*TEST 17       EXECUTE DATI, DATOB (HIGH BYTE) THRU MEMORY.
                                     ;*  EXECUTES THE INSTRUCTION 'MOVB R3,-(R2)' THROUGHOUT MEMORY.
                                     ;*  AN 'RTS R5' (CODE 205) IS PLACED AFTER THE 'MOVB' INSTRUCTION TO RETURN
                                     ;*  CONTROL TO THE MAIN PROGRAM FOR INSTRUCTION EXECUTION CHECKOUT.
                                     ;*  THIS IS AN EXAMPLE OF WHAT THIS TEST DOES IN RELATION TO MEMORY:
                                     ;*
                                     ;*              MEMORY       INSTRUCTION     CONTENTS OF MEMORY LOCATION
                                     ;*             LOCATION      PLACED THERE    AFTER INSTRUCTION EXECUTION
                                     ;*
                                     ;*   1ST PASS /  40000          110342          161342
                                     ;*  THRU TEST /  40002          000205          000205
                                     ;*
                                     ;*   2ND PASS /  40002          110342          161342
                                     ;*  THRU TEST /  40004          000205          000205
                                     ;*
                                     ;*      ETC., ETC., ETC.
                                     ;*
                                     ;*  R0 = DATA WRITTEN ON TOP OF IUT BY THE IUT (SHOULD BE).
                                     ;*  R1 = DATA READ FROM MEMORY (WAS).
                                     ;*  R2 = ADDRESS OF IUT/DATA.
                                     ;*  R3 = INSTRUCTION UNDER TEST (IUT).
                                     ;*  R4 = RTS R5 (CODE 205).
                                     ;*  R5 = BLOCK BOUNDARY BIT MASK.
                                     ;*****************************************************************
      011512                         TST17:
      011512  004567  006014                 JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
 2435 011516  012703  110342         DIDBH:  MOV     #110342,R3      ;GET 'MOVB R3,-(R2)' INSTRUCTION (IUT).
 2436 011522  012704  000205                 MOV     #205,   R4      ;GET 'RTS R5'
 2437 011526  012700  161342                 MOV     #161342,R0      ;SET UP S/B DATA AFTER EXECUTION.
 2438 011532  004467  001420                 JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
 2439 011536  005305                 1$:     DEC     R5              ;DO ALL ADDRESSES -1
 2440 011540  010322                         MOV     R3,     (R2)+   ;PUT IUT INTO FIRST LOC OF BLOCK.
 2441 011542  010412                 2$:     MOV     R4,     (R2)    ;PUT 'RTS R5' FOLLOWING IUT.
 2442 011544  004562  177776                 JSR     R5,     -2(R2)  ;GO EXECUTE THE IUT.
 2443 011550  005302                         DEC     R2              ;ADJUST R2 TO POINT TO MAUT.
 2444 011552  012201                         MOV     (R2)+,  R1      ;GET THE DATA FROM THE MEM ADR UNDER TEST.
 2445 011554  020001                         CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
      011556  001405                         BEQ     30053$          ;BRANCH OVER ERROR CALL IF GOOD DATA.
      011560  004767  005212         30052$: JSR     PC,     SPRNT3  ;SET UP VALUES FOR ERROR PRINTING.
      011564  004767  006254                 JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
      011570  000021                         .WORD   21              ;ERROR TYPE CODE.
      011572                         30053$:
 2446 011572  010322                         MOV     R3,     (R2)+   ;PUT THE IUT INTO THE NEXT LOCATION.
 2447 011574  077516                         SOB     R5,     2$      ;BRANCH IF MORE IN CURRENT BLOCK.
      011576  004767  001566                 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.
                        ;;*****************************************************************
011602                  TST20:
011602                  
011602  004567  005724          JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
011606  012703  005412  DIPDO:  MOV     #005412,R3      ;GET 'NEG (R2)' INSTRUCTION (IUT).
011612  012704  000205          MOV     #205,   R4      ;GET 'RTS R5'
011616  012700  172366          MOV     #172366,R0      ;SET UP S/B DATA AFTER EXECUTION.
011622  004467  001330          JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
011626  005305          1$:     DEC     R5              ;DO ALL ADDRESSES -1
011630  010322          2$:     MOV     R3,     (R2)+   ;PUT IUT INTO FIRST LOC OF BLOCK.
011632  010412                  MOV     R4,     (R2)    ;PUT 'RTS R5' FOLLOWING IUT.
011634  004542                  JSR     R5,     -(R2)   ;GO EXECUTE THE IUT.
011636  012201                  MOV     (R2)+,  R1      ;GET THE DATA FROM THE MEM ADR UNDER TEST.
011640  020001                  CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
011642  001405                  BEQ     30055$          ;BRANCH OVER ERROR CALL IF GOOD DATA.
011644  004767  005126  30054$: JSR     PC,     SPRNT3  ;SET UP VALUES FOR ERROR PRINTING.
011650  004767  006170          JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
011654  000021                  .WORD   21              ;ERROR TYPE CODE.
011656                  30055$:
011656  010322                  MOV     R3,     (R2)+   ;PUT THE IUT INTO THE NEXT LOCATION.
011660  077514                  SOB     R5,     2$      ;BRANCH IF MORE IN CURRENT BLOCK.
011662  004767  001502          JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
```

```
2513                    ;;*****************************************************************
                        ;*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.
                        ;*****************************************************************
     011666             TST21:
     011666  004567  005640          JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
2514 011672  012703  142242  DPDBL:  MOV     #142242,R3      ;GET BICB (R2)+,-(R2)' INSTRUCTION (IUT).
2515 011676  012704  000205          MOV     #205,   R4      ;GET 'RTS R5'
2516 011702  012700  142000          MOV     #142000,R0      ;SET UP S/B DATA AFTER EXECUTION.
2517 011706  004467  001244          JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2518 011712  005305          1$:     DEC     R5              ;DO ALL ADDRESSES -1
2519 011714  010322                  MOV     R3,     (R2)+   ;PUT IUT INTO FIRST LOC OF BLOCK.
2520 011716  010412          2$:     MOV     R4,     (R2)    ;PUT 'RTS R5' FOLLOWING IUT.
2521 011720  004542                  JSR     R5,     -(R2)   ;GO EXECUTE THE IUT.
2522 011722  012201                  MOV     (R2)+,  R1      ;GET THE DATA FROM THE MEM ADR UNDER TEST.
2523 011724  020001                  CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
     011726  001405                  BEQ     30057$          ;BRANCH OVER ERROR CALL IF GOOD DATA.
     011730  004767  005042  30056$: JSR     PC,     SPRNT3  ;SET UP VALUES FOR ERROR PRINTING.
     011734  004767  006104          JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
     011740  000021                  .WORD   21              ;ERROR TYPE CODE.
     011742          30057$:
2524 011742  010322                  MOV     R3,     (R2)+   ;PUT THE IUT INTO THE NEXT LOCATION.
2525 011744  077514                  SOB     R5,     2$      ;BRANCH IF MORE IN CURRENT BLOCK.
     011746  004767  001416          JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
```

```
                        ;*********************************************************
                        ;*TEST 22       EXECUTE DATI, DATI, DATIO, DATOB (HIGH BYTE) THRU MEMORY.
                        ;*  EXECUTES THE INSTRUCTION 'BISB (R2)+,(R2)' THROUGHOUT MEMORY.
                        ;*  AN 'RTS R5' (CODE 205) IS PLACED AFTER THE 'BISB' INSTRUCTION TO RETURN
                        ;*  CONTROL TO THE MAIN PROGRAM FOR INSTRUCTION EXECUTION CHECKOUT.
                        ;*  THIS IS AN EXAMPLE OF WHAT THIS TEST DOES IN RELATION TO MEMORY:
                        ;*
                        ;*              MEMORY       INSTRUCTION     CONTENTS OF MEMORY LOCATION
                        ;*             LOCATION      PLACED THERE    AFTER INSTRUCTION EXECUTION
                        ;*
                        ;*   1ST PASS /  40000          152212          157212
                        ;*  THRU TEST /  40002          000205          000205
                        ;*
                        ;*   2ND PASS /  40002          152212          157212
                        ;*  THRU TEST /  40004          000205          000205
                        ;*
                        ;*      ETC., ETC., ETC.
                        ;*
                        ;*  R0 = DATA WRITTEN ON TOP OF IUT BY THE IUT (SHOULD BE).
                        ;*  R1 = DATA READ FROM MEMORY (WAS).
                        ;*  R2 = ADDRESS OF IUT/DATA.
                        ;*  R3 = INSTRUCTION UNDER TEST (IUT).
                        ;*  R4 = RTS R5 (CODE 205).
                        ;*  R5 = BLOCK BOUNDARY BIT MASK.
                        ;*********************************************************
      011752            TST22:
      011752  004567  005554            JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
2552  011756  012703  152212  DPDBH:    MOV     #152212,R3      ;GET 'BISB (R2)+,(R2)  INSTRUCTION (IUT).
2553  011762  012704  000205            MOV     #205,   R4      ;GET 'RTS R5'
2554  011766  012700  157212            MOV     #157212,R0      ;SET UP S/B DATA AFTER EXECUTION.
2555  011772  004467  001160            JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2556  011776  005305            1$:     DEC     R5              ;DO ALL ADDRESSES -1
2557  012000  010322                    MOV     R3,     (R2)+   ;PUT IUT INTO FIRST LOC OF BLOCK.
2558  012002  010412            2$:     MOV     R4,     (R2)    ;PUT 'RTS R5' FOLLOWING IUT.
2559  012004  004542                    JSR     R5,     -(R2)   ;GO EXECUTE THE IUT.
2560  012006  005302                    DEC     R2              ;RESET R2 TO POINT TO IUT.
2561  012010  012201                    MOV     (R2)+,  R1      ;GET THE DATA FROM THE MEM ADR UNDER TEST.
2562  012012  020001                    CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
      012014  001405                    BEQ     30059$          ;BRANCH OVER ERROR CALL IF GOOD DATA.
      012016  004767  004754  30058$:  JSR     PC,     SPRNT3  ;SET UP VALUES FOR ERROR PRINTING.
      012022  004767  006016            JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
      012026  0000`1                    .WORD   21              ;ERROR TYPE CODE.
      012030            30059$:
2563  012030  010322                    MOV     R3,     (R2)+   ;PUT THE IUT INTO THE NEXT LOCATION.
2564  012032  077515                    SOB     R5,     2$      ;BRANCH IF MORE IN CURRENT BLOCK.
      012034  004767  001330            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.
                                ;;*****************************************************************
     012040                     TST23:
     012040  004567  005466             JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
2591 012044  004467  001106             JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2592 012050  010567  167716     100$:   MOV     R5,TEMP1        ;STORE TEST WORD COUNTER
2593 012054  010267  167710             MOV     R2,TEMP         ;SAVE BANK STARTING ADDRESS
2594 012060  005003             1$:     CLR     R3              ;CLEAR PASS COUNTER
2595 012062  012700  000377             MOV     #000377,R0      ;SETUP TO WRITE PATTERN
2596 012066  010022             2$:     MOV     R0,(R2)+        ;WRITE PATTERN
2597 012070  077502                     SOB     R5,2$           ;END OF 8K?
2598 012072  016705  167674             MOV     TEMP1,R5        ;RESTORE TEST WORD COUNTER FOR NEXT PASS
2599 012076  014201             3$:     MOV     -(R2),R1        ;GET DATA WRITEN
2600 012100  020001                     CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
     012102  001405                     BEQ     30061$          ;BRANCH OVER ERROR CALL IF GOOD DATA.
     012104  004767  004672     30060$: JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
     012110  004767  005730             JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
     012114  000010                     .WORD   10              ;ERROR TYPE CODE.
     012116                     30061$:
2601 012116  000300             4$:     SWAB    R0              ;SWAP BYTES OF DATA
2602 012120  010012                     MOV     R0,(R2) ;WRITE SWAPPED WORD
2603 012122  011201                     MOV     (R2),R1         ;GET DATA WRITEN
2604 012124  020001                     CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
     012126  001405                     BEQ     30063$          ;BRANCH OVER ERROR CALL IF GOOD DATA.
     012130  004767  004646     30062$: JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
     012134  004767  005704             JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
     012140  000010                     .WORD   10              ;ERROR TYPE CODE.
     012142                     30063$:
2605 012142  000300                     SWAB    R0              ;PUT DATA BACK TO ORINGINAL
2606 012144  005703                     TST     R3              ;IF ON PASS 0 OR PASS 3
2607 012146  001403                     BEQ     5$              ;WE ARE ADDRESSING DOWN
2608 012150  020327  000003             CMP     R3,#3           ;IF ON PASS 1 OR 2 GO TO
2609 012154  001011                     BNE     6$              ;UPWARD
2610 012156  077531             5$:     SOB     R5,3$           ;DONE A PASS? ,IF NOT THEN 3$
```

B6


```
 2611 012160  005203                    INC     R3              ;IF YES INCREMENT PASS COUNTER
 2612 012162  022703  000004            CMP     #4,R3           ;ARE WE DONE ALL PASSES FOR THIS 8K?
 2613 012166  001403                    BEQ     9$              ;IF YES BRANCH
 2614 012170  016705  167576            MOV     TEMP1,R5        ;RESTORE TEST COUNTER FOR NEXT PASS.
 2615 012174  000300                    SWAB    R0              ;ELSE SET UP NEW READ WORD
 2616 012176  000764                    BR      7$              ;GO TO START OF ADDRESS UP
 2617 012200  062702  000002      6$:   ADD     #2,R2           ;UPDATE TO NEXT ADDRESS
 2618 012204  005305                    DEC     R5              ;DONE A PASS?
 2619 012206  001411                    BEQ     8$              ;IF YES BRANCH
 2620 012210  011201              7$:   MOV     (R2),R1         ;GET DATA WRITTEN
 2621 012212  020001                    CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
      012214  001405                    BEQ     30065$          ;BRANCH OVER ERROR CALL IF GOOD DATA.
      012216  004767  004560    30064$: JSR     PC,     $PRNT2  ;SET UP VALUES FOR ERROR PRINTING.
      012222  004767  005616            JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
      012226  000010                    .WORD   10              ;ERROR TYPE CODE.
      012230                    30065$:
 2622 012230  000732                    BR      4$
 2623 012232  016705  167534      8$:   MOV     TEMP1,R5        ;RESTORE TEST WORD COUNTER FOR NEXT PASS.
 2624 012236  005203                    INC     R3              ;INCREMENT      PASS COUNTER
 2625 012240  000300                    SWAB    R0              ;SET UP NEW READ WORD
 2626 012242  020327  000002            CMP     R3,#2           ;ADDRESSING UP?
 2627 012246  001313                    BNE     3$              ;IF NO GO TO DOWN SEQUENCE
 2628 012250  016702  167514            MOV     TEMP,R2         ;IF YES RESET ADDRESS TO START
 2629 012254  000755                    BR      7$              ;GO TO UP SEQUENCE
 2630 012256  062702  040000      9$:   ADD     #40000,R2       ;TESTING ENDS WITH R2 = BOTTOM OF BANK
 2631                                                           ;ADD 8K FOR TEST START OF NEXT BANK
 2632 012262  042702  001000            BIC     #1000,R2        ;CLEAR POSSIBLE FIRST STARTING
 2633                                                           ;ADDRESS OF 1000 WHEN VECTOR AREA IS
 2634                                                           ;PROTECTED.
 2635 012266  004767  001076            JSR     PC,MMUP         ;UPDATE TO NEW BANK IF EXISTS
 2636                                                           ;RETURN TO 100$
 2637
 2650                              ;;*************************************  ***********************
                                   ;*TEST 24       WRITE CHECKERBOARD STARTING WITH '125252' DATA.
                                   ;* THESE TESTS WRITE A CHECKERBOARD THROUGHOUT MEMORY,STALL
                                   ;* FOR 2 SECONDS THEN CHECK PATTERN TO VERIFY DATA DID NOT
                                   ;* DETERIORATE BETWEEN REFRESH CYCLES.
                                   ;*
                                   ;*      R0=DATA WRITTEN INTO MEMORY(SHOULD BE)
                                   ;*      R1=DATA READ FROM MEMORY(WAS)
                                   ;*      R2=VIRTUAL ADDRESS
                                   ;*      R3=SMALL LOOP COUNTER FOR STALL
                                   ;*      R4=NUMBER OF TIMES SMALL LOOP DONE
                                   ;*      R5=BLOCK BOUNDARY BIT MASK.
                                   ;;*************************************************************
      012272                       TST24:
      012272  004567  005234            JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
 2651 012276  004467  000654            JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
 2652 012302  012700  125252            MOV     #125252 ,R0     ;SETUP DATA PATTERN
 2653 012306  010022              1$:   MOV     R0      ,(R2)+  ;WRITE A WORD
 2654 012310  005100                    COM     R0              ;COMPLEMENT DATA
 2655 012312  077503                    SOB     R5,     1$      ;BRANCH IF MORE IN CURRENT BLOCK
      012314  004767  001050            JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
 2656 012320  005003                    CLR     R3              ;SET UP COUNTER FOR STALL
 2657 012322  012704  000046            MOV     #46.    R4      ;DO LOOP 46 TIMES OR 2 SEC. TOTAL.
 2659 012326  005303              2$:   DEC     R3
 2659 012330  001376                    BNE     2$
```

```
                                        DEC     R4
                                        BNE     2$
                                        JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
                                        MOV     #125252,R0      ;INIT DATA FOR CHECKING
                                3$:
                                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
                                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
                                        BEQ     30067$          ;BRANCH OVER ERROR CALL IF GOOD DATA.
                                30066$: JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
                                        JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
                                        .WORD   6               ;ERROR TYPE CODE.
                                30067$:
                                        COM     R0
                                        SOB     R5,     3$      ;BRANCH IF MORE IN CURRENT BLOCK
                                        JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
                                ;;*****************************************************************
                                ;*TEST 25       WRITE CHECKERBOARD STARTING WITH 052525 DATA
                                ;;*****************************************************************
                                TST25:
                                        JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
                                        JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
                                        MOV     #052525,R0      ;SETUP DATA PATTERN
                                1$:     MOV     R0,     (R2)+   ;WRITE A WORD
                                        COM     R0
                                        SOB     R5,     1$      ;BRANCH IF MORE IN CURRENT BLOCK
                                        JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
                                        CLR     R3              ;SET COUNTER FOR LOOP
                                        MOV     #46,    R4      ;DO LOOP 46 TIMES OR 2 SEC. TOTAL
                                2$:     DEC     R3
                                        BNE     2$
                                        DEC     R4
                                        BNE     2$
                                        JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
                                        MOV     #052525,R0      ;INIT PATTERN FOR CHECKING
                                3$:
                                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
                                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
                                        BEQ     30069$          ;BRANCH OVER ERROR CALL IF GOOD DATA.
                                30068$: JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
                                        JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
                                        .WORD   6               ;ERROR TYPE CODE.
                                30069$:
                                        COM     R0
                                        SOB     R5,     3$      ;BRANCH IF MORE IN CURRENT BLOC<
                                        JSR     PC,     MMUP    ;FIND NEXT BLOC< AND LOOP TO 1$.
```

D6


```
                                                .SBTTL DONE:    RELOCATE PROGRAM AND REPEAT ALL TESTS.
012502                                          DONE:
012502  004567  005024                                JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
012506  005067  166446                          TST32: CLR     $TIMES          ;RESET ITERATION COUNTER FOR RESTARTING TEST.
012512  105067  166354                                 CLRB    $TSTNM          ;RESET TEST NUMBER.
012516  032777  000200  166404                         BIT     #SW07,  @SWR    ;CHECK FOR INHIBIT RELOCATION SWITCH.
012524  001037                                         BNE     $EOP            ;SKIP RELOCATION IF SWITCH SET.
012526  105767  166462                                 TSTB    $ENV            ;CHECK FOR APT11
012532  001403                                         BEQ     1$              ;BRANCH IF NOT APT11
012534  005767  166442                                 TST     $PASS           ;CHECK FOR  1ST PASS
012540  001026                                         BNE     6$              ;IF APT,DO NOT RELOCATE AFTER FIRST PASS
012542  026767  165274  165276              1$:      CMP     42,46           ;CHECK FOR ACT11
012550  001003                                         BNE     2$              ;BRANCH IF NOT ACT11
012552  005767  166424                                 TST     $PASS           ;CHECK FOR FIRST PASS
012556  001417                                         BEQ     6$              ;IF ACT,DO NOT RELOCATE ON FIRST PASS
012560  032767  000001  165746              2$:      BIT     #BIT0,PRGMAP    ;CHECK IF PROGRAM IN FIRST 8K
012566  001404                                         BEQ     4$              ;IF NOT IN FIRST 8K,RELOC TOP TO BOTTOM
                                                                               ;MUST BE XXDP OR STANDALONE
012570  004767  002154                                 JSR     PC,     RELTOP  ;RELOCATE PROGRAM TO TOP OF MEMORY.
012574  000167  174206                          3$:    JMP     START1          ;LOOP BACK AND RUN ALL TESTS AGAIN.

012600  004767  002460                          4$:    JSR     PC,     RELO    ;RELOCATE PROGRAM BACK TO FIRST 8K.
012604  005767  165232                                 TST     42              ;TEST FOR XXDP
012610  001402                                         BEQ     6$              ;IF NOT RUNNING UNDER MON. DONT
012612  004767  002646                          5$:    JSR     PC,     RESLDR  ;RESTORE LOADERS.
012616                                          6$:
012616  004567  007130                                 JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
012622  001171                                         .WORD   $CRLF           ;ADDRESS OF MESSAGE TO BE TYPED
```

```
2711                                    ;;**********************************************************************

                                        .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

012624                                  $EOP:
012624  000240                                  NOP
012626  005067  166326                        CLR     $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
012632  005267  166344                        INC     $PASS           ;;INCREMENT THE PASS NUMBER
012636  042767  100000  166336              BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
012644  005327                              DEC     (PC)+           ;;LOOP?
012646  000001                          $EOPCT: .WORD   1
012650  003041                              BGT     $DOAGN          ;;YES
012652  012737                              MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
012654  000001                          $ENDCT: .WORD   1
012656  012646                              $EOPCT
012660  004567  007066                        JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
012664  012760                              .WORD   $ENDMG          ;ADDRESS OF MESSAGE TO BE TYPED
012666  016746  166310                      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**.
012672  106746                              MFPS    -(SP)           ;PUT THE PROCESSOR STATUS ON THE STACK
012674  105066  000001                      CLRB    1(SP)           ;HIGH BYTE CLEARED TO INSURE KERNEL MODE
                                                                    ;ON PSW RETURN.
012700  004767  010042                      JSR     PC,     $TYPDS  ;GO TO THE SUBROUTINE
012704  004567  007042                      JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
012710  012775                              .WORD   $ENULL          ;ADDRESS OF MESSAGE TO BE TYPED
012712                                  $GET42:

012712  016700  165124                      MOV     42,     R0              ;;GET MONITOR ADDRESS
012716  001416                            BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
012720  000005                            RESET                   ;;CLEAR THE WORLD
012722  004710                          $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
012724  000240                            NOP                     ;;SAVE ROOM
012726  000240                            NOP                     ;;FOR
012730  000240                            NOP                     ;;ACT11
012732  023737  000042  000046            CMP     @#42,@#46       ;;ARE WE UNDER ACT11 OR XXDP
012740  001405                            BEQ     $DOAGN          ;;IF ACT11 THEN RESTART
012742  105737  001214                    TSTB    @#$ENV          ;;CHECK FOR APT11
012746  001002                            BNE     $DOAGN          ;;IF APT11 THEN RESTART
012750  004767  002576                    JSR     PC,     SAVLDR  ;;IF XXDP FIRST SAVE MONITOR
012754                                  $DOAGN:
012754  000167  174026                      JMP     START1          ;;RETURN*****
012760     015     012     105  $ENDMG: .ASCIZ  <15><12>/END PASS #/
012763     116     104     040
012766     120     101     123
012771     123     040     043
012774     000
012775     377     377     000  $ENULL: .BYTE   -1,-1,0          ;;NULL CHARACTER STRING
2712                                    .SBTTL SUBROUTINE AND TRAP ROUTINE SECTION.
2713                                    .SBTTL  MEMORY MANAGEMENT AND ADDRESSING SUBROUTINES.
2714                                    ;;**********************************************************************
2715                                    ;*  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.
                                        ;;*****************************************************************
2721 013000                             MMINIT:
     013000  012737  077406  172300             MOV     #200 1*400+UP+RW,@#KIPDR0       ;SET KIPDR0 = RW UP 200 BLOCKS
2722 013006  012737  077406  172302             MOV     #200-1*400+UP+RW,@#KIPDR1       ;SET KIPDR1 = RW UP 200 BLOCKS
2723 013014  012737  077406  172304             MOV     #200 1*400+UP+RW,@#KIPDR2       ;SET KIPDR2 = RW UP 200 BLOCKS
2724 013022  012737  077406  172306             MOV     #200-1*400+UP+RW,@#KIPDR3       ;SET KIPDR3 = RW UP 200 BLOCKS
2725 013030  005037  172310                     CLR     @#KIPDR4
2726 013034  005037  172312                     CLR     @#KIPDR5
2727 013040  105767  166060                     TSTB    $AUTOB                  ;;V IF TRUE XXDP CHAIN MODE
2728 013044  001005                             BNE     VMK                     ;;V DO NOT TOUCH KIPDR6 OR KIPDR7
2729 013046  005037  172314                     CLR     @#KIPDR6
2730 013052  012737  077406  172316             MOV     #200-1*400+UP+RW,@#KIPDR7       ;SET KIPDR7 = RW UP 200 BLOCKS
2731 013060  005037  172340             VMK:    CLR     @#KIPAR0        ;MAP PAR0 INTO BANK0
2732 013064  012737  000200  172342             MOV     #200, @#KIPAR1  ;MAP PAR1 INTO BANK1
2733 013072  005037  172344                     CLR     @#KIPAR2        ;MAP PAR2 INTO BANK0
2734 013076  005037  172346                     CLR     @#KIPAR3
2735 013102  005037  172350                     CLR     @#KIPAR4
2736 013106  005037  172352                     CLR     @#KIPAR5
2737 013112  105767  166006                     TSTB    $AUTOB                  ;;V IF TRUE XXDP CHAIN MODE
2738 013116  001005                             BNE     VMK1                    ;;V DO NOT TOUCH KIPAR6 OR KIPAR7
2739 013120  005037  172354                     CLR     @#KIPAR6
2740 013124  012737  177600  172356             MOV     #177600, @#KIPAR7       ;MAP PAR7 INTO I/O BANK
2741 013132  005767  165400             VMK1:   TST     MMAVA           ;IF 22 BIT ADR NOT AVAILABLE
2742 013136  100003                             BPL     1$              ;  THEN BRANCH
2743 013140  052737  000020  172516             BIS     #BIT4,@#SR3     ;  ELSE ENABLE 22 BIT ADDRESSING
2744 013146  012737  000001  177572     1$:     MOV     #1,@#SR0        ;ENABLE MEMORY MANAGEMENT
2745 013154  000207                             RTS     PC              ;RETURN
2746
2747
2748                                    ;;*****************************************************************
2749                                    ;*  MEMORY ADDRESS POINTER INITIALIZATION ROUTINES.
2750                                    ;;*****************************************************************
2751
2752 013156                             INITMM:
     013156  010046                             MOV     R0,-(SP)        ;;PUSH R0 ON STACK
     013160  010146                             MOV     R1,-(SP)        ;;PUSH R1 ON STACK
2753 013162  012767  100000  166612             MOV     #BIT15,PLUS1
2754 013170  005002                             CLR     R2              ;INITIALIZE ADDRESS
2755 013172  005767  165340                     TST     MMAVA
2756 013176  001410                             BEQ     2$
2757 013200  012700  000020                     MOV     #20,R0
2758 013204  016701  166606                     MOV     .BITPT,R1
2759 013210  005021                     1$:     CLR     (R1)+
2760 013212  077002                             SOB     R0,1$
2761 013214  005037  172344                     CLR     @#KIPAR2
2762 013220  012767  013362  166526     2$:     MOV     #INITEX,MMORE
2763 013226  066767  165300  166520             ADD     RELOCF,MMORE
2764 013234  012601                             MOV     (SP)+,R1        ;;POP STACK INTO R1
     013236  012600                             MOV     (SP)+,R0        ;;POP STACK INTO R0
2765 013240  004767  000124                     JSR     PC,MMUP
2766 013244  004767  004574                     JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
                                        ;FATAL ERROR
     013250  000007                             .WORD   7               ;ERROR TYPE CODE.
```

```
                                    HALT            ;FATAL ERROR!!!  NO MEM INDICATED IN MEMMAP ABOVE 8K!

                            INITDN:
013254  010046                        MOV     R0,-(SP)        ;;PUSH R0 ON STACK
013256  010146                        MOV     R1,-(SP)        ;;PUSH R1 ON STACK
013260  005002                        CLR     R2
013262  005767  165250                TST     MMAVA
013266  001414                        BEQ     2$
013270  012700  000020                MOV     #20,R0
013274  016701  166516                MOV     .BITPT,R1
013300  005021              1$:       CLR     (R1)+
013302  077002                        SOB     R0,1$
013304  012767  000001  166470        MOV     #BIT0,PLUS1
013312  005037  172344                CLR     @#KIPAR2
013316  000403                        BR      3$
013320  012767  000020  166366  2$:   MOV     #BIT4,BITPT
013326  012767  013362  166420  3$:   MOV     #INITEX,MMORE
013334  066767  165172  166412        ADD     RELOCF,MMORE
013342  012601                        MOV     (SP)+,R1        ;;POP STACK INTO R1
013344  012600                        MOV     (SP)+,R0        ;;POP STACK INTO R0
013346  004767  000362                JSR     PC,MMDOWN
013352  004767  004466                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
                            ;FATAL ERROR
013356  000007                        .WORD   7               ;ERROR TYPE CODE.
013360  000000                        HALT            ;FATAL ERROR!!!  NO MEM INDICATED IN MEMMAP ABOVE 8K!
013362  010467  166366        INITEX: MOV     R4,MMORE
013366  000204                        RTS     R4

                            ;;*****************************************************************
                            ;*  COMMON UPWARDS ADDRESSING ROUTINE
                            ;*  FINDS NEXT EXISTING 8K BANK AND UPDATES POINTERS.
                            ;*  GOES TO ADDRESS IN "MMORE" IF MORE BANKS,
                            ;*  DOES STRAIGHT EXIT WHEN ALL MEMORY HAS BEEN DONE.
                            ;;*****************************************************************

013370  012705  020000        MMUP:   MOV     #20000,R5       ;SET UP 8K TEST COUNTER
                            ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $CKSWR ROUTINE
                            ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
013374  106746                        MFPS    -(SP)           ;PUT THE PROCESSOR STATUS ON THE STACK
013376  105066  000001                CLRB    1(SP)           ;HIGH BYTE CLEARED TO INSURE KERNEL MODE
                                                              ;ON PSW RETURN.
013402  004767  005200                JSR     PC      $CKSWR  ;GO TO THE SUBROUTINE
013406  005767  165124                TST     MMAVA           ;CHECK FOR MEMORY MANAGEMENT
013412  001475                        BEQ     7$              ;IF MMAVA = 0 ,BRANCH TO NON XT MODE
013414  010046                        MOV     R0,-(SP)        ;;PUSH R0 ON STACK
013416  010146                        MOV     R1,-(SP)        ;;PUSH R1 ON STACK
013420  010346                        MOV     R3,-(SP)        ;;PUSH R3 ON STACK
013422  010446                        MOV     R4,-(SP)        ;;PUSH R4 ON STACK
013424  012702  040000                MOV     #40000,R2       ;MMU AVAIL. SET UP VA = 0,PAR2
013430  005767  166346                TST     PLUS1           ;CHECK IF FIRST TIME ENTRY
013434  001003                        BNE     100$            ;DO NOT ADD 8K TO PAR2
                                                              ;UPON FIRST ENTRY
013436  062737  000400  172344  1$:   ADD     #400,@#KIPAR2   ;ADD 8K TO PAR2
013444  016701  166346        100$:   MOV     .BITPT,R1
013450  012700  000020                MOV     #20,R0
013454  006367  166322                ASL     PLUS1
```

```
2811 013460  006121                  2$:    ROL    (R1)+              ;ROTATE TOTAL TABLE TO INSURE
2812 013462  077002                         SOB    R0,2$              ;ROTATION OF ONE BIT
2813 013464  103005                         BCC    101$               ;TESTING IS DONE WHEN CARRY BIT IS SET
2814 013466  012604                         MOV    (SP)+,R4           ;;POP STACK INTO R4
     013470  012603                         MOV    (SP)+,R3           ;;POP STACK INTO R3
     013472  012601                         MOV    (SP)+,R1           ;;POP STACK INTO R1
     013474  012600                         MOV    (SP)+,R0           ;;POP STACK INTO R0
2815 013476  000510                         BR     12$                ;DONE,READY TO EXIT
2816 013500  016703  166312            101$:  MOV    .BITPT,R3
2817 013504  016704  166302                  MOV    .TSTMAP,R4
2818 013510  012700  000020                  MOV    #20,R0
2819 013514  032324                  3$:    BIT    (R3)+,(R4)+
2820 013516  001002                         BNE    4$
2821 013520  077003                         SOB    R0,3$
2822 013522  000745                         BR     1$
2823 013524  013737  172344  172346  4$:    MOV    @#KIPAR2,@#KIPAR3
2824 013532  062737  000200  172346         ADD    #200,@#KIPAR3
2825 013540  012604                         MOV    (SP)+,R4           ;;POP STACK INTO R4
     013542  012603                         MOV    (SP)+,R3           ;;POP STACK INTO R3
     013544  012601                         MOV    (SP)+,R1           ;;POP STACK INTO R1
     013546  012600                         MOV    (SP)+,R0           ;;POP STACK INTO R0
2826 013550  005767  164762                  TST    MMAVA
2827 013554  100005                         BPL    5$
2828 013556  032767  100000  166166         BIT    #BIT15,BITPT+36
2829 013564  001005                         BNE    6$
2830 013566  000436                         BR     9$
2831 013570  032767  100000  166116  5$:    BIT    #BIT15,BITPT
2832 013576  001432                         BEQ    9$
2833 013600  012705  010000            6$:    MOV    #10000,R5
2834 013604  000427                         BR     9$
2835 013606  006367  166170            7$:    ASL    PLUS1
2836 013612  006167  166076                  ROL    BITPT
2837 013616  103440                         BCS    12$
2838 013620  036767  166070  165726         BIT    BITPT,TSTMAP
2839 013626  001003                         BNE    8$
2840 013630  062702  040000                  ADD    #40000,R2
2841 013634  000764                         BR     7$
2842 013636  032767  000010  166050  8$:    BIT    #BIT3,BITPT
2843 013644  001407                         BEQ    9$
2844 013646  012705  010000                  MOV    #10000,R5
2845 013652  005767  166116                  TST    FLG30K
2846 013656  001402                         BEQ    9$
2847 013660  062705  004000                  ADD    #4000,R5
2848 013664  032767  000001  166022  9$:    BIT    #BIT0,BITPT
2849 013672  001407                         BEQ    10$
2850 013674  066702  166062                  ADD    FSTADR,R2
2851 013700  005767  166056                  TST    FSTADR             ;FSTADR = STARTING ADDRESS
2852                                                                   ;     0 OR 1000 IF VECTOR AREA IS
2853                                                                   ;     PROTECTED.
2854 013704  001402                         BEQ    10$
2855 013706  162705  000400                  SUB    #400,R5            ;ADJUST ADDR COUNTER TO PROTECT 0-1000
2856 013712  016716  166036            10$:   MOV    MMORE,(SP)
2857 013716  000207                  11$:   RTS    PC
2858                                ;*  BEFORE FINAL EXIT, CHECK FOR ANY NON TRAP PARITY ERRORS.
2859 013720  005767  167446            12$:   TST    MPRX               ;CHECK FOR ANY PARITY REGISTERS PRESENT.
2860 013724  001402                         BEQ    13$                ;BR IF NONE.
2861 013726  004767  002106                  JSR    PC,CKPMER          ;CHECK FOR PARITY MEMORY ERRORS.
```

```
 2862 013732  000207                  13$:    RTS     PC              ;STRAIGHT RETURN.
 2863
 2864                                 ;;*****************************************************************
 2865                                 ;*  MEMORY DOWNWARDS ADDRESSING SUBROUTINE.
 2866                                 ;*  FINDS NEXT LOWER 8K BANK AND UPDATES POINTERS.
 2867                                 ;*  GOES TO ADDRESS IN "MMORE' IF MORE BANKS,
 2868                                 ;*  DOES STRAIGHT EXIT WHEN ALL MEMORY HAS BEEN DONE.
 2869                                 ;;*****************************************************************
 2870
 2871 013734  012705  020000          MM 'IN: MOV     #20000,R5       ;SET UP 8K TEST COUNTER
 2872                                 ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $CKSWR ROUTINE
                                      ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
      013740  106746                      MFPS    -(SP)           ;PUT THE PROCESSOR STATUS ON THE STACK
      013742  105066  000001              CLRB    1(SP)           ;HIGH BYTE CLEARED TO INSURE KERNEL MODE
                                                                   ;ON PSW RETURN.
      013746  004767  004634              JSR     PC,     $CKSWR  ;GO TO THE SUBROUTINE
 2873 013752  005767  164560              TST     MMAVA           ;CHECK FOR MEMORY MANAGEMENT
 2874 013756  001502                      BEQ     7$              ;IF MMAVA = 0 THEN BR TO NON KT MODE
 2875 013760  010046                      MOV     R0,-(SP)        ;;PUSH R0 ON STACK
      013762  010146                      MOV     R1,-(SP)        ;;PUSH R1 ON STACK
      013764  010346                      MOV     R3,-(SP)        ;;PUSH R3 ON STACK
      013766  010446                      MOV     R4,-(SP)        ;;PUSH R4 ON STACK
 2876 013770  162737  000400  172344  1$:     SUB     #400,@#KIPAR2   ;DOWN DATE TO NEXT 8K BANK
 2877 013776  016700  166014              MOV     .BITPT,R0
 2878 014002  062700  000040              ADD     #40,R0          ;SET UP TO MOVE POINTER
 2879 014006  012703  000020              MOV     #20,R3          ;ONE POSITION THRU 20 WORD TABLE
 2880 014012  006267  165764              ASR     PLUS1           ;GET INITIAL VALUE OF BIT
 2881 014016  006040                  2$:     ROR     -(R0)           ;MOVE THE POINTER
 2882 014020  077302                      SOB     R3,2$
 2883 014022  103005                      BCC     100$            ;CONTINUE TESTING UNTIL CARRY BIT SET
 2884 014024  012604                      MOV     (SP)+,R4        ;;POP STACK INTO R4
      014026  012603                      MOV     (SP)+,R3        ;;POP STACK INTO R3
      014030  012601                      MOV     (SP)+,R1        ;;POP STACK INTO R1
      014032  012600                      MOV     (SP)+,R0        ;;POP STACK INTO R0
 2885 014034  000517                      BR      11$             ;TESTING DONE,READY TO EXIT
 2886 014036  016700  165754          100$:   MOV     .BITPT,R0       ;THEN WE HAVE TESTED LAST BANK
 2887 014042  062700  000040              ADD     #40,R0          ;IF NO UNDERFLOW SET UP TO
 2888 014046  016701  165740              MOV     .TSTMAP,R1      ;SEE IF BANK SHOULD BE TESTED
 2889 014052  062701  000040              ADD     #40,R1
 2890 014056  012703  000020              MOV     #20,R3
 2891 014062  034041                  3$:     BIT     -(R0),-(R1)     ;CHECK IF BANK IS UNDER TEST
 2892 014064  001002                      BNE     4$              ;IF YES GO SET UP PARS
 2893 014066  077303                      SOB     R3,3$           ;IF NO CHECK ALL POSSIBLES
 2894 014070  000737                      BR      1$              ;IF NOT GO CHECK NEXT LOWER 8K
 2895 014072  013737  172344  172346  4$:     MOV     @#KIPAR2,@#KIPAR3;MAKE PAR2 AND PAR3 POINT TO SAME
 2896 014100  062737  000200  172346          ADD     #200,@#KIPAR3   ;MAKE KIPAR3 UPPER 4K
 2897 014106  012702  100000              MOV     #100000,R2      ;SET UP FIRST ADDRESS + 2
 2898 014112  012604                      MOV     (SP)+,R4        ;;POP STACK INTO R4
      014114  012603                      MOV     (SP)+,R3        ;;POP STACK INTO R3
      014116  012601                      MOV     (SP)+,R1        ;;POP STACK INTO R1
      014120  012600                      MOV     (SP)+,R0        ;;POP STACK INTO R0
 2899 014122  005767  164410              TST     MMAVA           ;CHECK FOR 22 BIT ADDRESSING
 2900 014126  100005                      BPL     5$              ;IF NO,CHECK 18 BIT MODE
 2901 014130  032767  100000  165614      BIT     #BIT15,BITPT+36 ;TESTING HIGHEST 22 BIT SEG.?
 2902 014136  001005                      BNE     6$              ;IF YES,DO ONLY 4K SEGMENT
 2903 014140  000442                      BR      9$              ;IF NO,DO 8K SEGMENT
 2904 014142  032767  100000  165544  5$:     BIT     #BIT15,BITPT    ;TESTING HIGHESTT 18 BIT SEGMENT
```

J6


```
2905 014150  001436                          BEQ     9$              ;IF NO ,DO 8K SEGMENT
2906 014152  012702  060000          6$:     MOV     #60000,R2       ;DO ONLY 4K SEGMENT
2907 014156  012705  010000                  MOV     #10000,R5       ;SET UP 4K WORD COUNTER
2908 014162  000431                          BR      9$              ;GET OVER NON KT ROUTINE
2909 014164  006267  165524          7$:     ASR     BITPT           ;POINT TO NEXT LOWEST BANK
2910 014170  103441                          BCS     11$             ;IF CARRY FOR LSB THEN DONE TESTING
2911 014172  036767  165516  165354          BIT     BITPT,TSTMAP    ;IS THIS BANK UNDER TEST?
2912 014200  001003                          BNE     8$
2913 014202  162702  040000                  SUB     #40000,R2
2914 014206  000766                          BR      7$
2915 014210  032767  000010  165476  8$:     BIT     #BIT3,BITPT
2916 014216  001413                          BEQ     9$
2917 014220  162702  020000                  SUB     #20000,R2
2918 014224  012705  010000                  MOV     #10000,R5
2919 014230  005767  165540                  TST     FLG30K
2920 014234  001404                          BEQ     9$
2921 014236  062702  010000                  ADD     #10000,R2
2922 014242  062705  004000                  ADD     #4000,R5
2923 014246  032767  000001  165440  9$:     BIT     #BIT0,BITPT
2924 014254  001405                          BEQ     10$
2925 014256  005767  165500                  TST     FSTADR          ;CHECK FIRST ADDRESS = 0 OR 1000
2926 014262  001402                          BEQ     10$
2927 014264  162705  000400                  SUB     #400,R5         ;PROTECT VECTOR AREA 0-1000 FROM
2928                                                                 ;BEING TESTED IF FSTADR =1000
2929 014270  016716  165460          10$:    MOV     MMORE,(SP)      ;SETUP RETURN FOR MORE TESTING
2930 014274  000207                  11$:    RTS     PC              ;RETURN
2931
```

```
                                .SBTTL  SUBROUTINES FOR ADDRESS AND WORSE CASE NOISE TESTS.
                                ;;*****************************************************************
                                ;*  SUBROUTINE TO CALCULATE PHYSICAL ADDRESS AND PUT IT IN R0 (BOTTOM 16 BITS).
                                ;;*****************************************************************
014276  005767  164234        PHYADR: TST     MMAVA           ;CHECK FOR MEM MGMT AVAILABLE
014302  001433                        BEQ     4$              ;BRANCH IF NO MEM MGMT
014304  010146                        MOV     R1,-(SP)        ;;PUSH R1 ON STACK
014306  010346                        MOV     R3,-(SP)        ;;PUSH R3 ON STACK
014310  010446                        MOV     R4,-(SP)        ;;PUSH R4 ON STACK
014312  010546                        MOV     R5,-(SP)        ;;PUSH R5 ON STACK
014314  010005                1$:     MOV     R0,R5           ;SAVE FOR PHYSICAL ADDR. ADDITION
014316  042700  017777                BIC     #17777,R0       ;CLEAR VIRTUAL ADDR BITS
014322  012701  000014                MOV     #12.,R1         ;SET UP ROTATE COUNTER
014326  006200                2$:     ASR     R0              ;TO SHIFT PAR POINTER
014330  077102                        SOB     R1,2$           ;INTO BITS 0-3
014332  012704  172340                MOV     #KIPAR0,R4      ;STORE START OF PARS
014336  060004                        ADD     R0,R4           ;USE PAR POINTER IN 0-3 AS INDEX
014340  011403                        MOV     (R4),R3         ;STORE CONTENTS OF PROPER PAR
014342  012701  000006                MOV     #6,R1           ;SET UP COUNTER TO
014346  006303                3$:     ASL     R3              ;CONVERT PAR CONTENTS
014350  077102                        SOB     R1,3$           ;TO A PHYSICAL "K" ADDRESS
014352  010500                        MOV     R5,R0           ;STORE VIRTUAL ADDRESS
014354  042700  160000                BIC     #160000,R0      ;CLEAR PAR POINTER
014360  060300                        ADD     R3,R0           ;ADD PHYSICAL "K" ADDRESS
                                                                ;TO OBTAIN 16 BIT PHYSICAL ADDRESS
014362  012605                        MOV     (SP)+,R5        ;;POP STACK INTO R5
014364  012604                        MOV     (SP)+,R4        ;;POP STACK INTO R4
014366  012603                        MOV     (SP)+,R3        ;;POP STACK INTO R3
014370  012601                        MOV     (SP)+,R1        ;;POP STACK INTO R1
014372  000207                4$:     RTS     PC              ;EXIT WITH PHYSICAL ADDR IN R0

                                ;;*****************************************************************
                                ;*  SUBROUTINE TO PUT BANK NUMBER INTO R0.
                                ;;*****************************************************************
014374  005000                BANKNO: CLR     R0              ;INIT R0
014376  010146                        MOV     R1,-(SP)        ;;PUSH R1 ON STACK
014400  010246                        MOV     R2,-(SP)        ;;PUSH R2 ON STACK
014402  010346                        MOV     R3,-(SP)        ;;PUSH R3 ON STACK
014404  010446                        MOV     R4,-(SP)        ;;PUSH R4 ON STACK
014406  016701  165404                MOV     .BITPT, R1      ;START OF BIT POINTER TABLE
014412  012703  000020                MOV     #20,    R3      ;SET UP BITPT TABLE COUNTER
014416  012702  000020        1$:     MOV     #20,    R2      ;SET UP ROTATE COUNTER FOR 1 WORD
014422  012104                        MOV     (R1)+,R4        ;GET WORD TO ROTATE BIT
014424  006204                2$:     ASR     R4              ;ROTATE BITPT WORD TO FIND BANK
014426  103410                        BCS     4$              ;IF CARRY SET ,THEN BANK FOUND
014430  105200                        INCB    R0              ;COUNT BANKS.
014432  001004                        BNE     3$              ;BR IF NOT OVERFLOW.
014434  004767  003404                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
                                ;FATAL ERROR
014440  000007                        .WORD   7               ;ERROR TYPE CODE.
014442  000000                        HALT                    ;FATAL ERROR!!!  NO POINTER FOUND.
014444  077211                3$:     SOB     R2,     2$      ;IF NOT FINISHED ROTATING WORD
                                                                ;GO BACK AND ROTATE AGAIN
014446  077315                        SOB     R3,     1$      ;IF NOT FINISHED BITPT TABLE
                                                                ;THEN ROTATE THROUGH NEXT WORD.
014450                          4$:
014450  012604                        MOV     (SP)+,R4        ;;POP STACK INTO R4
```

L6


```
        014452  012603                  MOV     (SP)+,R3        ;;POP STACK INTO R3
        014454  012602                  MOV     (SP)+,R2        ;;POP STACK INTO R2
        014456  012601                  MOV     (SP)+,R1        ;;POP STACK INTO R1
 2978   014460  000207                  RTS     PC              ;RETURN
 2979
 2980                           ;;*****************************************************************
 2981                           ;*  SUBROUTINE TO WRITE THE CONSTANT IN R0 INTO ALL OF MEMORY.
 2982                           ;;*****************************************************************
 2983   014462                  SETCON:
        014462  004467  176470          JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
 2984   014466  010022          1$:     MOV     R0,     (R2)+   ;MOV CONSTANT INTO MEMORY
 2985   014470  077502                  SOB     R5,     1$      ;BRANCH IF MORE IN CURRENT BLOCK
        014472  004767  176672          JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
 2986   014476  000207                  RTS     PC              ;RETURN
 2987
 2988                           ;;*****************************************************************
 2989                           ;* ROUTINE TO ROTATE 'C' BIT THROUGH A MEMORY LOCATION.
 2990                           ;;*****************************************************************
 2991   014500  106112          ROTATE: ROLB    (R2)            ;(R2)=177776 OR 000001
 2992   014502  106112                  ROLB    (R2)            ;(R2)=177775 OR 000002
 2993   014504  106112                  ROLB    (R2)            ;(R2)=177773 OR 000004
 2994   014506  106112                  ROLB    (R2)            ;(R2)=177767 OR 000010
 2995   014510  106112                  ROLB    (R2)            ;(R2)=177757 OR 000020
 2996   014512  106112                  ROLB    (R2)            ;(R2)=177737 OR 000040
 2997   014514  106112                  ROLB    (R2)            ;(R2)=177677 OR 000100
 2998   014516  106112                  ROLB    (R2)            ;(R2)=177777 OR 000000
 2999   014520  106122                  ROLB    (R2)+           ;(R2)=177577 OR 000200
 3000   014522  106112                  ROLB    (R2)            ;(R2)=177377 OR 000400
 3001   014524  106112                  ROLB    (R2)            ;(R2)=176777 OR 001000
 3002   014526  106112                  ROLB    (R2)            ;(R2)=175777 OR 002000
 3003   014530  106112                  ROLB    (R2)            ;(R2)=173777 OR 004000
 3004   014532  106112                  ROLB    (R2)            ;(R2)=167777 OR 010000
 3005   014534  106112                  ROLB    (R2)            ;(R2)=157777 OR 020000
 3006   014536  106112                  ROLB    (R2)            ;(R2)=137777 OR 040000
 3007   014540  106112                  ROLB    (R2)            ;(R2)=077777 OR 100000
 3008   014542  106122                  ROLB    (R2)+           ;(R2)=177777 OR 000000
 3009   014544  000207                  RTS     PC              ;RETURN
 3010
 3011                           ;;*****************************************************************
 3012                           ;*  SUBROUTINE TO WRITE 3 XOR 9 PATTERN INTO 256. WORD BLOCK.
 3013                           ;;*****************************************************************
 3014   014546  012704  000020  W3X9:   MOV     #16.,R4         ;EACH LOOP WRITES 256. WORDS
 3015
 3016   014552  010022          2$:     MOV     R0,(R2)+
 3017   014554  010022                  MOV     R0,(R2)+
 3018   014556  010022                  MOV     R0,(R2)+
 3019   014560  010022                  MOV     R0,(R2)+
 3020
 3021   014562  010322                  MOV     R3,(R2)+
 3022   014564  010322                  MOV     R3,(R2)+
 3023   014566  010322                  MOV     R3,(R2)+
 3024   014570  010322                  MOV     R3,(R2)+
 3025
 3026   014572  010022                  MOV     R0,(R2)+
 3027   014574  010022                  MOV     R0,(R2)+
 3028   014576  010022                  MOV     R0,(R2)+
 3029   014600  010022                  MOV     R0,(R2)+
```

M6


```
3030
3031 014602  010322                    MOV     R3,(R2)+
3032 014604  010322                    MOV     R3,(R2)+
3033 014606  010322                    MOV     R3,(R2)+
3034 014610  010322                    MOV     R3,(R2)+
3035
3036 014612  005304                    DEC     R4
3037 014614  001356                    BNE     2$
3038 014616  010046                    MOV     R0,     -(SP)   ;SAVE R0
     014620  010300                    MOV     R3,     R0      ;PUT R3 INTO R0
     014622  012603                    MOV     (SP)+,  R3      ;PUT SAVED R0 INTO R3
3039 014624  000207                    RTS     PC              ;RETURN
```

```
                                .SBTTL  RELOCATION SUBROUTINES.
                                ;;*****************************************************************
                                ;* ROUTINE TO RELOCATE PROGRAM CODE
                                ;;*****************************************************************
014626                          RELOC:
014626  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
014630  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
014632  010446                          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
014634  012502                  4$:     MOV     (R5)+, R2       ;GET FIRST LOCATION.
014636  012503                          MOV     (R5)+, R3       ;GET FIRST LOCATION OF DESTINATION.
014640  012704  020000                  MOV     #20000, R4      ;SET UP 8K COUNTER.
014644  012223                  1$:     MOV     (R2)+, (R3)+    ;MOV THE DATA.
014646  005304                          DEC     R4              ;COUNT THE WORDS.
014650  001375                          BNE     1$              ;BR IF MORE.
014652  012704  020000                  MOV     #20000, R4      ;RESET THE COUNTER.
014656  024243                  2$:     CMP     -(R2), -(R3)    ;CHECK THE DATA JUST MOVED.
014660  001417                          BEQ     3$              ;BR IF DATA OK.
014662  011267  164226                  MOV     (R2), $GDDAT    ;GET SOURCE DATA.
014666  011367  164224                  MOV     (R3), $BDDAT    ;GET DESTINATION DATA.
014672  010267  164212                  MOV     R2, $GDADR      ;GET SOURCE ADDRESS.
014676  010367  164210                  MOV     R3, $BDADR      ;GET DESTINATION ADDRESS.
014702  004767  003136                  JSR     PC, $ERROR      ;*** ERROR *** (GO TYPE A MESSAGE)
014706  000023                          .WORD   23              ;ERROR TYPE CODE.
014710  000000                          HALT                    ;FATAL ERROR!!!  RELOCATION FAILED.
014712  162705  000004                  SUB     #4, R5          ;ADJUST RETURN POINTER.
014716  000746                          BR      4$              ;GO BACK AND TRY AGAIN.
014720  005304                  3$:     DEC     R4              ;COUNT WORDS.
014722  001355                          BNE     2$              ;BR IF MORE.
014724  004567  005022                  JSR     R5, $PRINT      ;GO PRINT OUT THE FOLLOWING MESSAGE.
014730  024704                          .WORD   PRELOC          ;ADDRESS OF MESSAGE TO BE TYPED
                                                                ;"PROGRAM RELOCATED TO "
014732  010346                          MOV     R3, -(SP)       ;PUT THE DATA ON THE STACK.
014734  004767  006526                  JSR     PC, $TYPAD      ;DETERMINE THE PHYSICAL ADDRESS AND TYPE IT.
014740  012604                          MOV     (SP)+,R4        ;;POP STACK INTO R4
014742  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
014744  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
014746  000205                          RTS     R5              ;RETURN
                                ;;*****************************************************************
                                ;*  SUBROUTINE TO MOVE PROGRAM FROM BOTTOM TO TOP 8K BANK OF 128KW  MEMORY.
                                ;*      HIGHEST AVAILABLE BANK WITH KT WILL BE 700000 - 737776
                                ;*      HIGHEST AVAILABLE BANK NON-KT  WILL BE 100000 - 137776
                                ;;*****************************************************************
014750  022767  000001  163556  RELTOP: CMP     #1, PRGMAP      ;CHECK THAT THE PROGRAM IS NOW IN BANKS 0 AND 1.
014756  001404                          BEQ     1$              ;BR IF OK
014760  004767  003060                  JSR     PC, $ERROR      ;*** ERROR *** (GO TYPE A MESSAGE)
                                ;FATAL ERROR
014764  000007                          .WORD   7               ;ERROR TYPE CODE.
014766  000000                          HALT                    ;FATAL ERROR!!!  PROG SHOULD BE IN BANKS 0 AND 1
014770                          1$:
014770  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
014772  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
014774  005767  163536                  TST     MMAVA
015000  001441                          BEQ     10$
015002  012737  007600  172346          MOV     #7600, @#KIPAR3 ;SET PAR TO TOP OF MEM
015010  012700  100000                  MOV     #BIT15, R0      ;...HI 128K.
015014  162737  000400  172346  2$:     SUB     #400, @#KIPAR3          ;BACK DOWN ONE BANK.
015022  006000                          ROR     R0              ;..LO 64K.
```

```
3085 015024  103445                        BCS     90$
3086 015026  030067  164462                BIT     R0,     MEMMAP  ;CHECK FOR BANK EXISTS.
3087 015032  001770                        BEQ     2$              ;BR IF NO BANK FOUND.
3088 015034  013737  172346  172344  3$:   MOV     @#KIPAR3,@#KIPAR2 ;COPY PAR
3089 015042  162737  000200  172344  4$:   SUB     #200,   @#KIPAR2 ;BACK DOWN WITH LOW PAR.
3090 015050  030067  163460                BIT     R0,     PRGMAP  ;CHECK FOR CONFLICT.
3091 015054  001031                        BNE     90$             ;ABORT IF DESTINATION OVERLAYS SOURCE.
3092 015056  004567  177544                JSR     R5,     RELOC   ;GO RELOCATE PROGRAM.
3093 015062  000000                        .WORD   0               ;SOURCE FIRST ADDRESS.
3094 015064  040000                        .WORD   40000           ;DESTINATION FIRST ADDRESS.
3095 015066  013737  172344  172340        MOV     @#KIPAR2,@#KIPAR0 ;RELOCATE LO BANK
3096 015074  013737  172346  172342        MOV     @#KIPAR3,@#KIPAR1 ;RELOCATE HI BANK.
3097                              ;*  PROGRAM SHOULD NOW BE EXECUTING OUT OF LAST TWO BANKS OF MEMORY.
3098 015102  000461                        BR      30$             ;BR TO COMMON EXIT.
3099
3100 015104  012700  000010          10$:   MOV     #BIT3,  R0      ;SET BANK POINTER TO TOP OF MEM.
3101 015110  012701  140000                MOV     #140000,R1      ;SET ADDRESS POINTER TO TOP
3102 015114  162701  040000          11$:   SUB     #40000, R1      ;BACK DOWN ONE BANK.
3103 015120  006200                        ASR     R0              ;MOVE POINTER DOWN ONE BANK.
3104 015122  103406                        BCS     90$             ;BR IF OVERFLOW.
3105 015124  030067  164364                BIT     R0,     MEMMAP  ;CHECK IF THIS BANK EXISTS.
3106 015130  001771                        BEQ     11$             ;BR IF NON-EXISTANT BANK.
3107 015132  030067  163376                BIT     R0,     PRGMAP  ;CHECK FOR A PROGRAM CONFLICT.
3108 015136  001404                        BEQ     12$             ;BR IF NO CONFLICT.
3109 015140                          90$:
     015140  004767  002700                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
                                  ;FATAL ERROR
     015144  000007                        .WORD   7               ;ERROR TYPE CODE.
3110 015146  000000                        HALT                    ;FATAL ERROR!!!  NOT ENOUGH MEMORY??
3111 015150  010167  000006          12$:   MOV     R1,     13$     ;SET DATA FOR RELOCATION SUBROUTINE.
3112 015154  004567  177446                JSR     R5,     RELOC   ;GO RELOCATE THE PROGRAM TO TOP OF MEM.
3113 015160  000000                        .WORD   0               ;SOURCE STARTING ADDRESS.
3114 015162  000000          13$:          .WORD   0               ;DESTINATION STARTING ADDRESS.
3115 015164  010167  163342                MOV     R1,     RELOCF  ;SET RELOCATION FACTOR IN UNRELOCATED CODE.
3116 015170  060107                        ADD     R1,     PC      ;JUMP TO RELOCATED PROGRAM
3117                              ;*  PROGRAM NOW EXECUTING OUT OF TOP OF MEMORY.
3118 015172  060106                        ADD     R1,     SP      ;ADJUST THE STACK POINTER TO TOP OF MEMORY.
3119 015174  010167  163332                MOV     R1,     RELOCF  ;SET THE RELOCATION FACTOR.
3120 015200  060137  000004                ADD     R1,     @#ERRVEC ;ADJUST ERROR VECTOR.
3121 015204  060137  000024                ADD     R1,     @#PWRVEC ;ADJUST POWER FAIL VECTOR.
3122 015210  060137  000114                ADD     R1,     @#PARVEC ;ADJUST PARITY ERROR VECTOR.
3123 015214  060167  163710                ADD     R1,     SWR     ;ADJUST SOFTWARE SWITCH REGISTER.
3124 015220  060167  163706                ADD     R1      DISPLAY ;ADJUST SOFTWARE DISPLAY REGISTER.
3125 015224  062701  002004          14$:   ADD     #RADTAB,R1      ;POINT TO THE RELATIVE RELOCATION TABLE.
3126 015230  066721  163276          15$:   ADD     RELOCF, (R1)+   ;ADD RELOCATION FACTOR TO ADDRESSES IN TABLE.
3127 015234  005721                  16$:   TST     (R1)+           ;CHECK FOR INTERUM TERMINATOR.
3128 015236  001776                        BEQ     16$             ;BR SO AS TO NOT MODIFY ZERO.
3129 015240  024127  177777                CMP     -(R1),  #-1     ;CHECK FOR END OF TABLE.
3130 015244  001371                        BNE     15$             ;BR IF MORE IN TABLE.
3131 015246  010067  163262          30$:   MOV     R0,     PRGMAP  ;SET NEW PROGRAM MAP...LO 64K.
3132 015252  012601                        MOV     (SP)+,R1        ;;POP STACK INTO R1
     015254  012600                        MOV     (SP)+,R0        ;;POP STACK INTO R0
3133 015256  066716  163250                ADD     RELOCF, (SP)    ;ADJUST RETURN ADDRESS.
3134 015262  000207                        RTS     PC              ;RETURN
3135
3136                              ;;*****************************************************************
3137                              ;*  SUBROUTINE TO RELOCATE PROGRAM BACK TO BANKS 0 AND 1.
```

```
3138                                    ;;*****************************************************************
3139 015264  032767  000001  163242 RELO:   BIT     #1,     PRGMAP  ;CHECK FOR PROGRAM ALREADY IN BANKS 0 OR 1.
3140 015272  001404                          BEQ     1$              ;BR IF NO CONFLICT.
3141 015274  004767  002544                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
                                        ;FATAL ERROR
     015300  000007                          .WORD   7               ;ERROR TYPE CODE.
3142 015302  000000                          HALT                    ;FATAL ERROR!!!  PROGRAM ALREADY IN BANKS 0 OR 1!!!!
3143 015304  005767  163226          1$:     TST     MMAVA           ;CHECK FOR MEM MGMT.
3144 015310  001417                          BEQ     10$             ;BR IF NO MEMMGMT.
3145 015312  005037  172344                  CLR     @#KIPAR2        ;SET PAR 2 TO BANK 0.
3146 015316  012737  000200  172346          MOV     #200,   @#KIPAR3 ;SET PAR 3 TO BANK 1.
3147 015324  004567  177276                  JSR     R5,     RELOC   ;GO MOVE 8K INTO BANKS 0 AND 1.
3148 015330  000000                          .WORD   0               ;SOURCE STARTING ADDRESS.
3149 015332  040000                          .WORD   40000           ;DESTINATION STARTING ADDRESS.
3150 015334  005037  172340                  CLR     @#KIPAR0        ;RESTORE PAR 0 TO BANK0.
3151 015340  012737  000200  172342          MOV     #200,   @#KIPAR1 ;RESTORE PAR 1 TO BANK 1.
3152                                    ;*  PROGRAM IS NOW EXECUTING OUT OF BANKS 0 AND 1.
3153 015346  000440                          BR      30$             ;BR TO COMMON EXIT.
3154
3155 015350  016746  163156          10$:    MOV     RELOCF, -(SP)   ;PUT RELOCATION FACTOR ONTO THE STACK.
3156 015354  011667  000004                  MOV     (SP),   20$     ;SET DATA FOR RELOC SUBROUTINE.
3157 015360  004567  177242                  JSR     R5,     RELOC   ;GO MOVE THE PROGRAM BACK TO BANKS 0 AND 1.
3158 015364  000000                  20$:    .WORD   0               ;SOURCE STARTING ADDRESS.
3159 015366  000000                          .WORD   0               ;DESTINATION STARTING ADDRESS.
3160 015370  161607                          SUB     (SP),   PC      ;JUMP TO RELOCATED PROGRAM.
3161                                    ;*  THE PROGRAM IS NOW EXECUTING OUT OF BANKS 0 AND 1.
3162 015372  161606                          SUB     (SP),   SP      ;RESET THE STACK POINTER.
3163 015374  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
3164 015376  012700  002004                  MOV     #RADTAB,R0      ;SET UP POINTER TO RELATIVE ADDRESS TABLE.
3165
3166 015402  166620  000002          21$:    SUB     2(SP),  (R0)+   ;RESET ADDRESSES TO UNRELOCATED VALUES.
3167 015406  005720                  22$:    TST     (R0)+           ;CHECK FOR TERMINATORS.
3168 015410  001776                          BEQ     22$             ;BR OVER TERMINATORS.
3169 015412  024027  177777                  CMP     -(R0),  #-1     ;CHECK FOR END OF TABLE INDICATOR.
3170 015416  001371                          BNE     21$             ;BR IF MORE ADDRESSES IN TABLE.
3171 015420  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
3172 015422  161637  000004                  SUB     (SP),   @#ERRVEC ;ADJUST ERROR VECTOR.
3173 015426  161637  000024                  SUB     (SP),   @#PWRVEC ;ADJUST POWER FAIL VECTOR.
3174 015432  161637  000114                  SUB     (SP),   @#PARVEC ;ADJUST PARITY ERROR VECTOR.
3175 015436  161667  163466                  SUB     (SP),   SWR     ;ADJUST SOFTWARE SWITCH REGISTER.
3176 015442  161667  163464                  SUB     (SP),   DISPLAY ;ADJUST SOFTWARE DISPLAY REGISTER.
3177 015446  162616                  23$:    SUB     (SP)+,  (SP)    ;ADJUST RETURN ADDRESS.
3178 015450  005067  163056          30$:    CLR     RELOCF          ;RESET RELOCATION FACTOR.
3179 015454  012767  000001  163052          MOV     #1,     PRGMAP  ;SET PROGRAM MAP TO POINT TO BANKS 0 AND 1.
3180 015462  000207                          RTS     PC              ;RETURN.
3181
3182                                    ;;*****************************************************************
3183                                    ;* THIS SUBROUTINE MOVES THE LOADER AREA BACK TO THE "TOP" OF MEMORY FROM
3184                                    ;*  WHENCE IT CAME.  THE LOADER AREA IS SAVED AT THE END OF THE 8K OF
3185                                    ;*  PROGRAM CODE WHEN THE PROGRAM IS INITIALLY RUN.
3186                                    ;;*****************************************************************
3187 015464  016700  164020          RESLDR: MOV     LMAD,   R0      ;CHECK IF THE LOADERS WERE SAVED.
3188 015470  001004                          BNE     1$              ;BR IF LOADER AREA WAS SAVED.
3189 015472  004767  002346                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
                                        ;FATAL ERROR
     015476  000007                          .WORD   7               ;ERROR TYPE CODE.
3190 015500  000000                          HALT                    ;FATAL ERROR!!!  CAN'T RESTORE LOADER AREA IF IT WASN'T SAVED.
```

```
3191 015502  005767  163030          1$:   TST   MMAVA           ;CHECK FOR MEM MGMT.
3192 015506  001402                      BEQ   2$               ;SKIP IF NO MEM MGMT.
3193 015510  005037  177572              CLR   @#SR0            ;DISABLE MEM MGMT.
3194 015514  012701  040000          2$:   MOV   #40000, R1     ;GET END OF 8K, ASSUME PROG NOT RELOCATED.
3195 015520  012702  003000              MOV   #1536., R2       ;GET COUNTER.
3196 015524  014140                  3$:   MOV   (R1), -(R0)    ;MOVE THE LOADER AREA.
3197 015526  005302                      DEC   R2               ;COUNT HOW LONG THE AREA IS.
3198 015530  001375                      BNE   3$               ;BR IF NOT MORE TO MOVE.
3199 015532  005067  163752              CLR   LMAD             ;CLEAR MONITOR SAVED FLAG
3200 015536  005767  162774              TST   MMAVA            ;CHECK FOR MEM MGMT.
3201 015542  001402                      BEQ   4$               ;BR IF NO MEM MGMT.
3202 015544  005237  177572              INC   @#SR0            ;ENABLE MEM MGMT.
3203 015550  000207                  4$:   RTS   PC               ;RETURN.
3204
3205                                 ;*  ROUTINE TO SAVE THE LOADERS AT THE END OF 8K.
3206 015552  005767  163732          SAVLDR: TST   LMAD         ;CHECK IF LOADERS HAVE BEEN SAVED ALREADY.
3207 015556  001024                      BNE   4$               ;BRANCH IF ALREADY SAVED
3208 015560  012700  040000              MOV   #40000, R0       ;GET END OF 8K
3209 015564  010001                      MOV   R0, R1           ;GET END OF 8K
3210 015566  012737  015600  000004      MOV   #2$, @#ERRVEC    ;SET UP TIMEOUT VECTOR
3211 015574  011020                  1$:   MOV   (R0), (R0)+    ;SEARCH FOR END OF MEMORY
3212 015576  000776                      BR    1$               ;KEEP SEARCHING
3213 015600  022626                  2$:   CMP   (SP)+, (SP)+   ;RESTORE STACK POINTER
3214 015602  012737  023430  000004      MOV   #ERRTRP,@#ERRVEC ;RESET TIMEOUT VECTOR.
3215 015610  010046                      MOV   R0, -(SP)        ;SAVE LAST MEMORY ADDRESS (CONTIGUOUS)
3216 015612  012702  003000              MOV   #1536., R2       ;SET UP WORD COUNTER
3217 015616  014041                  3$:   MOV   -(R0), -(R1)   ;SAVE THE LOADERS
3218 015620  005302                      DEC   R2               ;COUNT THE WORDS
3219 015622  001375                      BNE   3$               ;BRANCH IF MORE WORDS
3220 015624  012667  163660              MOV   (SP)+, LMAD      ;SAVE LAST MEMORY ADDRESS
3221 015630  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.
                                        ;;*****************************************************************
3229 015632  011667  163254    PESRV:  MOV     (SP),   $BDADR  ;GET PC OF INSTRUCTION WHICH CAUSED ERROR.
3230 015636  004567  004110            JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
     015642  024643                    .WORD   UNEXPT          ;ADDRESS OF MESSAGE TO BE TYPED
                                                                ;"UNEXPECTED MEMORY PARITY TRAP."
3231 015644  010146                    MOV     R1,-(SP)        ;;PUSH R1 ON STACK
     015646  010346                    MOV     R3,-(SP)        ;;PUSH R3 ON STACK
3232 015650  016703  164150            MOV     .MPRX,  R3      ;GET POINTER TO PARITY REGISTERS.
3233 015654  005733            1$:     TST     @(R3)+          ;CHECK THE PARITY REG FOR AN ERROR FLAG.
3234 015656  100415                    BMI     3$              ;BR IF THIS REGISTER SHOWS THE ERROR.
3235 015660  005713                    TST     (R3)            ;CHECK FOR TABLE TERMINATOR.
3236 015662  001374                    BNE     1$              ;BR IF MORE REGISTERS.
3237 015664  004767  002154            JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
                                        ;***ERROR***    NO REGISTER INDICATED ERROR
     015670  000024                    .WORD   24              ;ERROR TYPE CODE.
3238 015672  000417                    BR      4$              ;EXIT
3239 015674  005713            2$:     TST     (R3)            ;CHECK FOR TABLE TERMINATOR.
3240 015676  001415                    BEQ     4$              ;BR IF NO MORE PARITY REGISTERS.
3241 015700  005733                    TST     @(R3)+          ;CHECK THE PARITY REG FOR AN ERROR FLAG.
3242 015702  100374                    BPL     2$              ;BR IF NO ERROR FLAG.
3243 015704  004567  004042            JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
     015710  024734                    .WORD   MTOE            ;ADDRESS OF MESSAGE TO BE TYPED
                                                                ;"MORE THAN ONE ERROR FOUND."
3244 015712                    3$:
     015712  004767  001010    30070$: JSR     PC,     SPRNTQ  ;SET UP VALUES FOR ERROR PRINTING.
     015716  004767  002122            JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
     015722  000025                    .WORD   25              ;ERROR TYPE CODE.
3245 015724  004767  000262            JSR     PC,     PSCAN   ;GO SCAN MEMORY FOR BAD PARITY.
3246 015730  000761                    BR      2$              ;GO LOOK FOR MORE ERRORS.
3247 015732                    4$:
     015732  012603                    MOV     (SP)+,R3        ;;POP STACK INTO R3
     015734  012601                    MOV     (SP)+,R1        ;;POP STACK INTO R1
3248 015736  000002                    RTI                     ;RETURN.
3249
3250                                    ;;*****************************************************************
3251                                    ;ROUTINE TO ENABLE PARITY ERROR ACTION ON MA/MF PARITY MEMORIES
3252                                    ;THIS ROUTINE IS MEANT TO CATCH UNEXPECTEDS
3253                                    ;;*****************************************************************
3254 015740  005767  165426    MAMF:   TST     MPRX            ;CHECK IF ANY PARITY REGISTERS EXIST.
3255 015744  001434                    BEQ     MAMF2           ;EXIT IF NO PARITY REGISTERS.
3256 015746  032777  000100  163154    BIT     #SW6,   @SWR    ;CHECK FOR INHIBIT PARITY ERROR DETECTION.
3257 015754  001030                    BNE     MAMF2           ;EXIT IF NO PARITY ERROR DETECTION.
3258 015756  005767  162550            TST     RELOCF          ;CHECK IF PROGRAM RELOCATED OUT OF BANK 0.
3259 015762  001410                    BEQ     SETAE           ;BR IF PROG IN BANK 0.
3260 015764  032777  000040  163136    BIT     #SW5,   @SWR    ;CHECK IF VECTORS PROTECTED.
3261 015772  001004                    BNE     SETAF           ;BR IF VECTOR AREA PROTECTED.
3262 015774  026727  163762  001000    CMP     FSTADR, #1000   ;CHECK FOR STARTING ADDRESS ABOVE THE VECTORS.
3263 016002  103415                    BLO     MAMF2           ;EXIT IF VECTORS EXPOSED TO TESTING.
3264
3265 016004  016737  164022  000114  SETAE:  MOV     .PESRV, @#PARVEC ;SET PARITY ERROR TRAP VECTOR
3266 016012  005037  000116            CLR     @#PARVEC+2      ;PRIORITY LEVEL 0 ON TRAP
3267 016016  010346                    MOV     R3,-(SP)        ;;PUSH R3 ON STACK
```

```
3268 016020  016703  164000            MAMF1:  MOV     .MPRX,  R3      ;GET PARITY REGISTER TABLE POINTER.
3269 016024  052733  000001                    BIS     #AE,    @(R3)+  ;SET ACTION ENABLE BIT IN PARITY REG
3270 016030  005713                            TST     (R3)            ;CHECK FOR END OF TABLE.
3271 016032  001374                            BNE     MAMF1           ;BR IF MORE PARITY REGISTERS.
3272 016034  012603                            MOV     (SP)+,R3        ;;POP STACK INTO R3
3273 016036  000207                    MAMF2:  RTS     PC              ;RETURN.
3274
3275                                    ;;*****************************************************************
3276                                    ;*  SUBROUTINE TO CHECK PARITY REGISTERS FOR ERRORS THAT DIDN'T TRAP.
3277                                    ;;*****************************************************************
3278 016040                            CHGG2:
3279 016040  005767  165326            CKPMER: TST     MPRX            ;CHECK IF ANY PARITY REGISTERS EXIST.
3280 016044  001437                            BEQ     4$              ;BR IF NO PARITY REGISTERS.
3281 016046  032777  000100  163054            BIT     #SW6,   @SWR    ;CHECK FOR INHIBIT PARITY ERROR CHECKING.
3282 016054  001033                            BNE     4$              ;BR IF PARITY ERROR CHECKING INHIBITED.
3283 016056  010346                            MOV     R3,-(SP)        ;;PUSH R3 ON STACK
3284 016060  016703  163740                    MOV     .MPRX,  R3      ;GET PARITY REG TABLE POINTER.
3285 016064  005733                    1$:     TST     @(R3)+          ;CHECK THE PARITY REG FOR AN ERROR FLAG.
3286 016066  100023                            BPL     3$              ;BR IF NO ERROR
3287 016070  032773  000001  177776            BIT     #BIT0,  @-2(R3) ;CHECK IF A TRAP SHOULD HAVE OCCURRED.
3288 016076  001410                            BEQ     2$              ;BR IF NO ACTION ENABLE.   CHGG2
3289 016100  004767  000622            30071$: JSR     PC,     SPRNTQ  ;SET UP VALUES FOR ERROR PRINTING.
     016104  004767  001734                    JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
     016110  000026                            .WORD   26              ;ERROR TYPE CODE.
3290 016112  000411                            BR      3$
3291 016114  004767  000072                    JSR     PC,     PSCAN   ;GO SCAN ALL MEMORY FOR PARITY ERRORS.
3292 016120                            2$:
     016120  004767  000602            30072$: JSR     PC,     SPRNTQ  ;SET UP VALUES FOR ERROR PRINTING.
     016124  004767  001714                    JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
     016130  000027                            .WORD   27              ;ERROR TYPE CODE.
3293 016132  004767  000054                    JSR     PC,     PSCAN   ;GO SCAN ALL MEMORY FOR PARITY ERRORS.
3294 016136  005713                    3$:     TST     (R3)            ;CHECK FOR TABLE TERMINATOR.
3295 016140  001351                            BNE     1$              ;BR IF MORE.
3296 016142  012603                            MOV     (SP)+,R3        ;;POP STACK INTO R3
3297 016144  000207                    4$:     RTS     PC              ;RETURN.
3298
3299                                    ;;*****************************************************************
3300                                    ;* SUBROUTINE TO VERIFY THAT THE PARITY CSR DOES CONTROL THE BANK
3301                                    ;* UNDER TEST. MATCH THE BANK UNDER TEST WITH THE PARITY MAP TABLE.
3302                                    ;* RETURN IF MEM BANK HAS BEEN MAPPED IN PARITY MAP TABLE.
3303                                    ;* RETURN+2 IF ERROR,MEM BANK HAS NOT BEEN MAPPED.
3304                                    ;;*****************************************************************
3305
3306 016146                            PARMAT:
     016146  010046                            MOV     R0,-(SP)        ;;PUSH R0 ON STACK
     016150  010146                            MOV     R1,-(SP)        ;;PUSH R1 ON STACK
     016152  010446                            MOV     R4,-(SP)        ;;PUSH R4 ON STACK
3307 016154  012700  000020                    MOV     #20,R0          ;COUNT OF PARITY MAP TABLE
3308 016160  016701  163632                    MOV     .BITPT,R1       ;MEM BANK UNDER TEST POINTER
3309 016164  010304                            MOV     R3,R4           ;STORE PARITY MAP TABLE
3310 016166  005724                            TST     (R4)+           ;MOVE PAST CSR ADDRESS TO MAP TABLE
3311 016170  032124                    1$:     BIT     (R1)+,(R4)+     ;CHECK FOR TABLE MATCH
3312 016172  001003                            BNE     2$              ;BR IF MATCH MADE IN TABLES
3313 016174  005300                            DEC     R0              ;DO NEXT WORD IN TABLE
3314 016176  001374                            BNE     1$              ;BR TO NEXT WORD IN TABLE
3315 016200  005725                            TST     (R5)+           ;NO MATCH MADE,RETURN +2
3316 016202                            2$:
```

```
        016202  012604                          MOV     (SP)+,R4        ;;POP STACK INTO R4
        016204  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
        016206  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
  3317  016210  000205                          RTS     R5
  3318
  3319
  3320                                  ;;*****************************************************************
  3321                                  ;* AFTER A PARITY ERROR IS ENCOUNTERED THIS SUBROUTINE IS USED
  3322                                  ;* TO SCAN ALL PARITY MEMORY PER THE PMEMAP TABLE LOOKING FOR BYTE LOCATIONS
  3323                                  ;* CAUSING THE PARITY ERROR.  UPON FINDING THE LOCATION(S) A MESSAGE
  3324                                  ;* WILL BE TYPED AND THE LOCATION WILL BE REWRITTEN IN ORDER TO RESTORE GOOD PARITY.
  3325                                  ;*      STORAGE USED:
  3326                                  ;*              R0      = BANK POINTER
  3327                                  ;*              R1      = PARITY CSR TABLE POINTER (MPRX)
  3328                                  ;*              R2      = ADDRESS POINTER
  3329                                  ;*              R3      = PMEMAP TABLE ENTRY
  3330                                  ;*              R4      = ERROR DETECT FLAG
  3331                                  ;*              TEMP    = TEMPORARY STORAGE
  3332                                  ;;*****************************************************************
  3333  016212                          PSCAN:
        016212  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
        016214  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
        016216  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
        016220  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
        016222  010446                          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
  3334  016224  004567  003522                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
        016230  025000                          .WORD   SCANM           ;ADDRESS OF MESSAGE TO BE TYPED
                                                                        ;"SCANNING MEMORY FOR BAD PARITY"
  3335  016232  004767  000434                  JSR     PC,CLRPAR       ;GO INITIALIZE PARITY CSR'S
  3336  016236  013746  000114                  MOV     @#114,-(SP)     ;;PUSH @#114 ON STACK
        016242  013746  000116                  MOV     @#116,-(SP)     ;;PUSH @#116 ON STACK
  3337  016246  012737  000116  000114          MOV     #116,@#114      ;SET UP PARITY VECTOR TO
  3338  016254  005037  000116                  CLR     @#116           ;HALT IF ANOTHER PARITY ERROR OCCURS
  3339  016260  005004                          CLR     R4              ;INITIALIZE ERROR DETECT FLAG
  3340  016262  005002                          CLR     R2              ;INITIALIZE ADDRESS POINTER
  3341  016264  012700  000001                  MOV     #BIT0,R0        ;INITIALIZE BANK POINTER
  3342  016270  016701  163520                  MOV     .PMEMAP,R1      ;POINT TO PMEMAP TABLE
  3343  016274  005767  162236                  TST     MMAVA           ;IF MEMORY MANAGEMENT NOT AVAILABLE
  3344  016300  001413                          BEQ     1$              ;  THEN BRANCH
  3345  016302  013746  172344                  MOV     @#KIPAR2,-(SP)          ;;PUSH @#KIPAR2 ON STACK
        016306  013746  172346                  MOV     @#KIPAR3,-(SP)          ;;PUSH @#KIPAR3 ON STACK
  3346  016312  005037  172344                  CLR     @#KIPAR2        ;INITIALIZE PAR2 TO 0
  3347  016316  012737  000200  172346          MOV     #200,@#KIPAR3   ;INITIALIZE PAR3
  3348  016324  012702  040000                  MOV     #40000,R2       ;INITIALIZE ADDRESS POINTER TO MAP THRU PAR2
  3349
  3350  016330  030011                  1$:     BIT     R0,(R1)         ;IF 8K BANK DOES EXIST
  3351  016332  001003                          BNE     2$              ;  THEN BRANCH
  3352  016334  062702  040000                  ADD     #40000,R2       ;  ELSE UPDATE ADDRESS TO NEXT 8K BANK
  3353  016340  000475                          BR      10$             ;    AND BRANCH TO SEE IF SHOULD CONTINUE TESTING
  3354
  3355  016342  111267  163422          2$:     MOVB    (R2),TEMP       ;READ THE BYTE LOCATION
  3356  016346  016703  163452                  MOV     .MPRX,R3        ;POINT TO PARITY CSR TABLE
  3357
  3358  016352  005733                  3$:     TST     @(R3)+          ;IF NO ERROR FLAG SET IN CSR
  3359  016354  100024                          BPL     5$              ;  THEN BRANCH
  3360  016356  005704                          TST     R4              ;  ELSE IF NOT FIRST ERROR
  3361  016360  001003                          BNE     4$              ;     THEN BRANCH
```

```
3362 016362  005367  162514                    DEC     $ERTTL          ;     ELSE ADJUST ERROR COUNT
3363 016366  005204                            INC     R4              ;       AND SET ERROR DETECT FLAG
3364
3365 016370                            4$:
     016370  004767  000332            30073$: JSR     PC,     SPRNTQ  ;SET UP VALUES FOR ERROR PRINTING.
     016374  004767  001444                    JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
     016400  000030                            .WORD   30              ;ERROR TYPE CODE.
3366 016402  111212                            MOVB    (R2),(R2)       ;REWRITE LOCATION TO CLEAR BAD PARITY
3367 016404  005053                            CLR     @-(R3)          ;CLEAR ERROR FLAG IN PARITY CSR
3368 016406  105712                            TSTB    (R2)            ;REREAD LOCATION TO SEE IF ERROR CLEARED
3369 016410  005733                            TST     @(R3)+          ;IF ERROR CLEARED
3370 016412  001405                            BEQ     5$              ;  THEN BRANCH
3371 016414  004567  003332                    JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
     016420  025042                            .WORD   PEWNC           ;ADDRESS OF MESSAGE TO BE TYPED
                                                                        ;"PARITY ERROR WILL NOT CLEAR"
3372 016422  005073  177776                    CLR     @-2(R3)         ;  CLEAR PARITY CSR
3373
3374 016426  005713                    5$:     TST     (R3)            ;IF NOT DONE WITH TABLE
3375 016430  001350                            BNE     3$              ;  THEN TRY NEXT CSR FOR ERROR
3376 016432  005202                            INC     R2              ;  ELSE UPDATE ADDRESS POINTER
3377 016434  005767  162076                    TST     MMAVA           ;IF MEMORY MANAGEMENT AVAILABLE
3378 016440  001013                            BNE     7$              ;  THEN BRANCH
3379 016442  005767  163326                    TST     FLG30K          ;  ELSE IF NOT 30K SYSTEM
3380 016446  001404                            BEQ     6$              ;    THEN BRANCH
3381 016450  022702  170000                    CMP     #170000,R2      ;    ELSE IF 30K I/O PAGE
3382 016454  001467                            BEQ     14$             ;      THEN ALL DONE CHECKING MEMORY
3383 016456  000423                            BR      9$              ;      ELSE TRY SOMEMORE
3384
3385 016460  022702  160000            6$:     CMP     #160000,R2      ;IF 28K I/O PAGE
3386 016464  001463                            BEQ     14$             ;  THEN ALL DONE CHECKING MEMORY
3387 016466  000417                            BR      9$              ;  ELSE TRY SOMEMORE
3388
3389 016470  022702  060000            7$:     CMP     #60000,R2       ;IF NOT POSSIBLY THE I/O PAGE
3390 016474  001014                            BNE     9$              ;  THEN BRANCH AND TRY SOMEMORE
3391 016476  005767  162034                    TST     MMAVA           ;  ELSE IF 22 BIT ADDRESSING AVAILABLE
3392 016502  100405                            BMI     8$              ;    THEN BRANCH
3393 016504  022737  007600  172346            CMP     #7600,@#KIPAR3  ;    ELSE IF 128K I/O PAGE
3394 016512  001444                            BEQ     13$             ;      THEN ALL DONE
3395 016514  000404                            BR      9$              ;      ELSE TRY SOMEMORE
3396
3397 016516  022737  177600  172346    8$:     CMP     #177600,@#KIPAR3        ;IF 2M I/O PAGE
3398 016524  001437                            BEQ     13$             ;  THEN ALL DONE CHECKING
3399
3400 016526  032702  037777            9$:     BIT     #MASK8K,R2      ;IF NOT DONE WITH THIS 8K BANK
3401 016532  001303                            BNE     2$              ;  THEN TRY ANOTHER ADDRESS
3402
3403 016534  006300                    10$:    ASL     R0              ;IF NOT DONE WITH 128K BANK
3404 016536  001012                            BNE     11$             ;  THEN BRANCH
3405 016540  062701  000002                    ADD     #2,R1           ;  ELSE POINT TO NEXT PMEMAP ENTRY
3406 016544  016700  163244                    MOV     .PMEMAP,R0      ;START OF TABLE
3407 016550  062700  000040                    ADD     #40,R0          ;END OF TABLE
3408 016554  020001                            CMP     R0,R1           ;IF DONE WITH TABLE
3409 016556  001422                            BEQ     13$             ;  THEN ALL DONE
3410 016560  012700  000001                    MOV     #BIT0,R0        ;  ELSE REINIT BANK POINTER
3411
3412 016564  005767  161746            11$:    TST     MMAVA           ;IF KT AVAILABLE
3413 016570  001004                            BNE     12$             ;  THEN BRANCH
```

```
3414 016572  032700  000020            BIT     #BIT4,R0        ;  ELSE IF AT 32K LIMIT
3415 016576  001016                    BNE     14$             ;    THEN ALL DONE CHECKING
3416 016600  000653                    BR      1$              ;    ELSE TRY SOMEMORE
3417
3418 016602  062737  000400  172344 12$:    ADD     #400,@#KIPAR2   ;UPDATE PARS TO MAP TO NEXT
3419 016610  062737  000400  172346        ADD     #400,@#KIPAR3   ;  8K BANK
3420 016616  012702  040000                MOV     #40000,R2       ;REINIT ADDRESS
3421 016622  000642                    BR      1$              ;AND TRY SOMEMORE
3422
3423 016624                        13$:
     016624  012637  172346                MOV     (SP)+,@#KIPAR3          ;;POP STACK INTO @#KIPAR3
     016630  012637  172344                MOV     (SP)+,@#KIPAR2          ;;POP STACK INTO @#KIPAR2
3424
3425 016634  005704                14$:    TST     R4              ;IF ERROR FOUND
3426 016636  001003                    BNE     15$             ;  THEN BRANCH
3427 016640  004567  003106            JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
     016644  024204                    .WORD   NOPES           ;ADDRESS OF MESSAGE TO BE TYPED
3428
3429 016646                        15$:
     016646  012637  000116                MOV     (SP)+,@#116     ;;POP STACK INTO @#116
     016652  012637  000114                MOV     (SP)+,@#114     ;;POP STACK INTO @#114
     016656  012604                        MOV     (SP)+,R4        ;;POP STACK INTO R4
     016660  012603                        MOV     (SP)+,R3        ;;POP STACK INTO R3
     016662  012602                        MOV     (SP)+,R2        ;;POP STACK INTO R2
     016664  012601                        MOV     (SP)+,R1        ;;POP STACK INTO R1
     016666  012600                        MOV     (SP)+,R0        ;;POP STACK INTO R0
3430
3431 016670  000207                16$:    RTS     PC              ;RETURN
3432
3433
3434                               ;;*****************************************************************
3435                               ;ROUTINE TO CLEAR ALL PARITY REGISTERS PRESENT
3436                               ;;*****************************************************************
3437 016672                        CLRPAR:
     016672  010346                        MOV     R3,-(SP)        ;;PUSH R3 ON STACK
3438 016674  016703  163124                MOV     .MPRX,  R3      ;GET PARITY REGISTER TABLE POINTER.
3439 016700  005713                1$:     TST     (R3)            ;CHECK FOR THE TABLE TERMINATOR.
3440 016702  001402                    BEQ     2$              ;BR IF DONE ALL PARITY REGISTERS.
3441 016704  005033                    CLR     @(R3)+          ;CLEAR THE PARITY REGISTER.
3442 016706  000774                    BR      1$              ;BR FOR MORE
3443 016710                        2$:
     016710  012603                        MOV     (SP)+,R3        ;;POP STACK INTO R3
3444 016712  000207                    RTS     PC              ;RETURN.
3445
3446
3447                               .SBTTL  SUBROUTINES TO SET UP DATA FOR ERROR PRINTOUT ROUTINE.
3448                               ;;*****************************************************************
3449                               ;*  THESE ROUTINES ARE USED TO TRANSFER DATA TO COMMON TAG AREA (.$CMTAG)
3450                               ;*  FOR ERROR PRINTOUT BY .$ERROR & .$ERRTYP ROUTINES FROM **SYSMAC**.
3451                               ;;*****************************************************************
3452 016714  010267  162170        SPRNT:  MOV     R2,     $GDADR  ;SAVE THE ADDRESS UNDER TEST.
3453 016720  005067  162170                CLR     $GDDAT          ;SHOULD BE DATA IS "0".
3454 016724  000435                    BR      SPRNTB
3455
3456 016726  014367  162216        SPRNTQ: MOV     -(R3),  $TMP0   ;GET THE PARITY REGISTER ADDRESS.
3457 016732  013367  162214                MOV     @(R3)+, $TMP1   ;GET THE CONTENTS OF THE PARITY REG.
3458 016736  016767  163026  162152        MOV     TEMP,$GDDAT
```

```
                        MOV     R2,$GDADR
                        BR      SPREND

SPRNTP: MOV     (R3),   $TMPO   ;GET THE PARITY REGISTER ADDRESS.
SPRNTO: MOV     R2,     $GDADR  ;GET THE MEMORY ADDRESS BEING TESTED
                        BR      SPRNTA          ;BR TO COMMON SECTION.

SPRNT1: MOV     R2,     $GDADR  ;GET THE MEMORY ADDRESS BEING TESTED
                        DEC     $GDADR          ;ADJUST IT FOR PRINTOUT.
                        BR      SPRNTA          ;BR TO COMMON SECTION.

SPRNT3: MOV     R3,     $TMPO   ;GET THE DATA IN R3.
SPRNT2: MOV     R2,     $GDADR  ;GET THE MEMORY ADDRESS BEING TESTED
                        SUB     #2,     $GDADR  ;ADJUST IT FOR PRINTOUT.
SPRNTA: MOV     R0,     $GDDAT  ;GET WHAT THE DATA SHOULD BE
SPRNTB: MOV     R1,     $BDDAT  ;GET WHAT THE DATA WAS
SPREND: RTS     PC              ;RETURN TO ENTER ERROR ROUTINES


;;*****************************************************************
;*  THIS ROUTINE WILL TYPE OUT A MAP OF MEMORY BANKS.
;*  UPON ENTERING THIS ROUTINE R0 WILL CONTAIN THE ADDRESS
;*  OF THE TABLE TO BE TYPED OUT.  THE ROUTINE REQUIRES THAT THE
;*  TABLE CONTAIN 16 DECIMAL WORDS.
;*      STORAGE USED:
;*              R0      = MAP TABLE ADDRESS
;*              R1      = BANK POINTER
;*              R2      = CONSECUTIIVE ZERO BANK FLAG
;*              R3      = HIGH MEMORY ADR ... LO 16 BITS
;*              R4      = HIGH MEMORY ADR ... HI 6 BITS
;*              LMEMLO  = LOW MEMORY ADR ... LO 16 BITS
;*              LMEMHI  = LOW MEMORY ADR ... HI 6 BITS
;*              BNKTAB  = ADDRESS OF TABLE TO BE PROCESSED
;*              ENDBKT  = LAST ADDRESS + 2 OF TABLE TO BE PROCESSED
;*              FLG30K  = 30K MEMORY FLAG
;;*****************************************************************
TYPMAP: MOV     R0,@#BNKTAB     ;SAVE ADDRESS OF TABLE
        MOV     #40,@#ENDBKT    ;LOAD END BANK TABLE WITH 40
        ADD     R0,@#ENDBKT     ;GET LAST ADR + 2 INTO ENDBKT
1$:     TST     (R0)            ;ANY MEMORY IN THIS TABLE ENTRY
        BNE     2$              ;  THEN GO PRINT OUT LOCATIONS
        TST     (R0)+           ;  ELSE POINT TO NEXT TABLE ENTRY
        CMP     @#ENDBKT,R0     ;IF NOT END OF TABLE
        BNE     1$              ;  THEN TRY AGAIN
        JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
        .WORD   NOMEM           ;ADDRESS OF MESSAGE TO BE TYPED
                                ;"NO MEMORY FOUND."
        JMP     13$             ;EXIT

2$:     MOV     @#BNKTAB,R0     ;RESTORE MAP TABLE ADDRESS
        MOV     R1,-(SP)        ;;PUSH R1 ON STACK
        MOV     R2,-(SP)        ;;PUSH R2 ON STACK
        MOV     R3,-(SP)        ;;PUSH R3 ON STACK
        MOV     R4,-(SP)        ;;PUSH R4 ON STACK
        CLR     LMEMLO          ;INIT LOW MEM ADR ... LO 16 BITS
        CLR     LMEMHI          ;INIT LOW MEM ADR ... HI 6 BITS
        CLR     TEMP            ;INIT LAST BANK IN MAP FLAG
```

```
3511 017122  012703  177777          MOV     #-1,R3          ;INIT HIGH MEM ADR ... LO 16 BITS
3512 017126  010304                  MOV     R3,R4           ;INIT HIGH MEM ADR ... HI 6 BITS
3513 017130  012702  000001          MOV     #1,R2           ;INIT CONSECUTIVE ZERO BANK FLAG
3514
3515 017134  012701  000001  3$:     MOV     #BIT0,R1        ;INIT BANK POINTER
3516
3517 017140  030110          4$:     BIT     R1,(R0)         ;IF THIS 8K BANK IS NOT PRESENT
3518 017142  001405                  BEQ     5$              ;  THEN GO PRINTOUT MAP
3519 017144  062703  040000          ADD     #40000,R3       ;  ELSE UPDATE HIGH MEM TO TOP OF THIS BANK
3520 017150  005504                  ADC     R4              ;
3521 017152  005002                  CLR     R2              ;CLEAR CONSECUTIVE ZERO BANK FLAG
3522 017154  000541                  BR      11$             ;GO CHECK FOR NEXT BANK
3523
3524 017156  005702          5$:     TST     R2              ;IF CONSECUTIVE ZERO BANK
3525 017160  001123                  BNE     10$             ;  THEN DO NOT PRINT MESSAGE
3526 017162  005202                  INC     R2              ;  ELSE UPDATE CONSECUTIVE ZERO BANK FLAG
3527 017164  022703  177777          CMP     #177777,R3      ;IF NOT 32K, 128K OR 2M BOUNDARY
3528 017170  001023                  BNE     9$              ;  THEN GO PRINT MESSAGE
3529 017172  005767  161340          TST     MMAVA           ;  ELSE IF 22BIT ADDRESSING
3530 017176  100413                  BMI     7$              ;    THEN GO WORK ON 22 BIT ADDRESSING
3531 017200  001006                  BNE     6$              ;    OR GO WORK ON 18 BIT ADDRESSING
3532 017202  005767  162566          TST     FLG30K          ;    ELSE IF NOT 30K SYSTEM
3533 017206  001412                  BEQ     8$              ;      THEN GO ADJ TO 28K BOUNDARY
3534 017210  162703  010000          SUB     #10000,R3       ;      ELSE ADJ TO 30K BOUNDARY
3535 017214  000411                  BR      9$              ;AND GO PRINT
3536
3537 017216  022704  000003  6$:     CMP     #3,R4           ;IF NOT 128K BOUNDARY
3538 017222  001006                  BNE     9$              ;  THEN GO PRINT
3539 017224  000403                  BR      8$              ;  ELSE GO ADJ TO 124K BOUNDARY
3540
3541 017226  022704  000077  7$:     CMP     #77,R4          ;IF NOT 2M BOUNDARY
3542 017232  001002                  BNE     9$              ;  THEN GO PRINT
3543
3544 017234  162703  020000  8$:     SUB     #20000,R3       ;ADJ HIGH MEMORY TO BELOW I/O PAGE
3545
3546 017240          9$:
     017240  010346                  MOV     R3,-(SP)        ;;PUSH R3 ON STACK
     017242  010446                  MOV     R4,-(SP)        ;;PUSH R4 ON STACK
     017244  016746  000252          MOV     LMEMLO,-(SP)    ;;PUSH LMEMLO ON STACK
     017250  016746  000250          MOV     LMEMHI,-(SP)    ;;PUSH LMEMHI ON STACK
3547 017254  004567  002472          JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
     017260  024005                  .WORD   FROM            ;ADDRESS OF MESSAGE TO BE TYPED
3548 017262  006303                  ASL     R3              ;SHIFT BIT 15 INTO CARRY
3549 017264  006104                  ROL     R4              ;SHIFT BIT 15 TO ALIGN WITH BITS 17-16
3550 017266  006003                  ROR     R3              ;RESTORE BITS 14-0
3551 017270  006367  000226          ASL     LMEMLO          ;SHIFT BIT 15 INTO CARRY
3552 017274  006167  000224          ROL     LMEMHI          ;SHIFT BIT 15 TO ALIGN WITH BITS 17 16
3553 017300  006067  000216          ROR     LMEMLO          ;RESTORE BITS 14-0
3554 017304  016746  000214          MOV     LMEMHI,-(SP)    ;;SAVE LMEMHI FOR TYPEOUT
                                                             ;;TYPE ADDRESS BITS 21-15
                                     ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOS ROUTINE
                                     ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
     017310  106746                      MFPS    -(SP)           ;PUT THE PROCESSOR STATUS ON THE STACK
     017312  105066  000001              CLRB    1(SP)           ;HIGH BYTE CLEARED TO INSURE KERNEL MODE
                                                                 ;ON PSW RETURN.
     017316  004767  003656              JSR     PC,     $TYPOS  ;GO TO THE SUBROUTINE
     017322     003                      .BYTE   3               ;;TYPE 3 DIGIT(S)
```

```
      017323    000                          .BYTE   0                ;;SUPPRESS LEADING ZEROS
 3555 017324  016746  000172                MOV     LMEMLO,(SP)      ;;SAVE LMEMLO FOR TYPEOUT
                                                                      ;;TYPE ADDRESS BITS 14-0
                                   ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOS ROUTINE
                                   ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
      017330  106746                         MFPS    -(SP)            ;PUT THE PROCESSOR STATUS ON THE STACK
      017332  105066  000001                 CLRB    1(SP)            ;HIGH BYTE CLEARED TO INSURE KERNEL MODE
                                                                      ;ON PSW RETURN.
      017336  004767  003636                 JSR     PC,     $TYPOS   ;GO TO THE SUBROUTINE
      017342     005                         .BYTE   5                ;;TYPE 5 DIGIT(S)
      017343     001                         .BYTE   1                ;;TYPE LEADING ZEROS
 3556 017344  004567  002402                JSR     R5,     $PRINT   ;GO PRINT OUT THE FOLLOWING MESSAGE.
      017350  024015                         .WORD   T0               ;ADDRESS OF MESSAGE TO BE TYPED
 3557 017352  010446                         MOV     R4,-(SP)         ;;SAVE R4 FOR TYPEOUT
                                                                      ;;TYPE ADDRESS BITS 21-15
                                   ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOS ROUTINE
                                   ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
      017354  106746                         MFPS    -(SP)            ;PUT THE PROCESSOR STATUS ON THE STACK
      017356  105066  000001                 CLRB    1(SP)            ;HIGH BYTE CLEARED TO INSURE KERNEL MODE
                                                                      ;ON PSW RETURN.
      017362  004767  003612                 JSR     PC,     $TYPOS   ;GO TO THE SUBROUTINE
      017366     003                         .BYTE   3                ;;TYPE 3 DIGIT(S)
      017367     000                         .BYTE   0                ;;SUPPRESS LEADING ZEROS
 3558 017370  010346                         MOV     R3,-(SP)         ;;SAVE R3 FOR TYPEOUT
                                                                      ;;TYPE ADDRESS BITS 14-0
                                   ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOS ROUTINE
                                   ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
      017372  106746                         MFPS    -(SP)            ;PUT THE PROCESSOR STATUS ON THE STACK
      017374  105066  000001                 CLRB    1(SP)            ;HIGH BYTE CLEARED TO INSURE KERNEL MODE
                                                                      ;ON PSW RETURN.
      017400  004767  003574                 JSR     PC,     $TYPOS   ;GO TO THE SUBROUTINE
      017404     005                         .BYTE   5                ;;TYPE 5 DIGIT(S)
      017405     001                         .BYTE   1                ;;TYPE LEADING ZEROS
 3559 017406  012667  000112                MOV     (SP)+,LMEMHI     ;;POP STACK INTO LMEMHI
      017412  012667  000104                MOV     (SP)+,LMEMLO     ;;POP STACK INTO LMEMLO
      017416  012604                         MOV     (SP)+,R4         ;;POP STACK INTO R4
      017420  012603                         MOV     (SP)+,R3         ;;POP STACK INTO R3
 3560 017422  005767  162342                TST     TEMP             ;HAS LAST BIT IN MAP PRINTED?
 3561 017426  001030                        BNE     14$              ;YES,EXIT TO STACK CLEANUP
 3562
 3563 017430  062703  040000          10$:  ADD     #40000,R3        ;UPGRADE TO TOP OF THIS BANK
 3564 017434  005504                        ADC     R4               ;
 3565 017436  010367  000060                MOV     R3,LMEMLO        ;UPDATE LMEM TO THIS BANK
 3566 017442  010467  000056                MOV     R4,LMEMHI        ;
 3567 017446  062767  000001  000046        ADD     #1,LMEMLO        ;GET TO NEW LOW ADDRESS OF NEXT BANK
 3568 017454  005567  000044                ADC     LMEMHI           ;
 3569
 3570 017460  006301                  11$:  ASL     R1               ;POINT TO NEXT 8K BANK
 3571 017462  001226                        BNE     4$               ;BRANCH IF MORE TO TEST IN THIS MAP ENTRY
 3572
 3573 017464  005720                  12$:  TST     (R0)+            ;UPDATE POINTER TO NEXT TABLE ENTRY
 3574 017466  023700  017530                CMP     @#ENDBKT,R0      ;IF NOT DONE WITH TABLE
 3575 017472  001220                        BNE     3$               ;  THEN TRY AGAIN
 3576 017474  005760  177776                TST     -2(R0)           ;CHECK IF LAST BANK AVAILABLE
 3577 017500  100003                        BPL     14$              ;IF ZERO,TYPMAP PRINT COMPLETED
 3578 017502  005267  162262                INC     TEMP             ;ELSE SET LAST BANK PRINT FLAG
 3579 017506  000623                        BR      5$               ;PRINT FOR LAST BANK
```

```
 3580 017510                            14$:
      017510  012604                            MOV     (SP)+,R4        ;;POP STACK INTO R4
      017512  012603                            MOV     (SP)+,R3        ;;POP STACK INTO R3
      017514  012602                            MOV     (SP)+,R2        ;;POP STACK INTO R2
      017516  012601                            MOV     (SP)+,R1        ;;POP STACK INTO R1
 3581
 3582 017520  000207                    13$:    RTS     PC              ;RETURN TO CALLER
 3583
 3584 017522  000000                    LMEMLO: .WORD   0               ; LOW MEMORY ADR ... LO 16 BITS
 3585 017524  000000                    LMEMHI: .WORD   0               ; LOW MEMORY ADR ... HI 6 BITS
 3586 017526  000000                    BNKTAB: .WORD   0               ; ADDRESS OF TABLE TO BE PROCESSED
 3587 017530  000000                    ENDBKT: .WORD   0               ; LAST ADDRESS + 2 OF TABLE
 3588                                                                   ; TO BE PROCESSED.
 3589
 3596                                   .SBTTL  SCOPE HANDLER ROUTINE
                                        ;;*************************************'*********************
                                        ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
                                        ;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
                                        ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
                                        ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
                                        ;*SW14=1         LOOP ON TEST
                                        ;*SW11=1         INHIBIT ITERATIONS
                                        ;*SW09=1         LOOP ON ERROR
                                        ;*SW08=1         LOOP ON TEST IN SWR<4:0>
                                        ;*CALL
                                        ;*      SCOPE           ;;SCOPE=IOT
      017532                            $SCOPE:
                                        ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $CKSWR ROUTINE
                                        ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
      017532  106746                            MFPS    -(SP)           ;PUT THE PROCESSOR STATUS ON THE STACK
      017534  105066  000001                    CLRB    1(SP)           ;HIGH BYTE CLEARED TO INSURE KERNEL MODE
                                                                         ;ON PSW RETURN.
      017540  004767  001042                    JSR     PC,     $CKSWR  ;GO TO THE SUBROUTINE
      017544  010516                            MOV     R5,     (SP)    ;PUT RETURN PC ONTO STACK, SIMULATE JSR PC.
      017546  032777  040000  161354    1$:     BIT     #BIT14,@SWR     ;;LOOP ON PRESENT TEST?
      017554  001117                            BNE     $OVER           ;;YES IF SW14=1
                                        ;#####START OF CODE FOR THE XOR TESTER#####
      017556  000416                    $XTSTR: BR      6$              ;;IF RUNNING ON THE "XOR" TESTER CHANGE
                                                                         ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
      017560  013746  000004                    MOV     @#ERRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
      017564  012737  017604  000004            MOV     #5$,@#ERRVEC    ;;SET FOR TIMEOUT
      017572  005737  177060                    TST     @#177060        ;;TIME OUT ON XOR?
      017576  012637  000004                    MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
      017602  000466                            BR      $SVLAD          ;;GO TO THE NEXT TEST
      017604  022626                    5$:     CMP     (SP)+,(SP)+     ;;CLEAR THE STACK AFTER A TIME OUT
      017606  012637  000004                    MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
      017612  000426                            BR      7$              ;;LOOP ON THE PRESENT TEST
      017614                            6$:;#####END OF CODE FOR THE XOR TESTER#####
      017614  032777  000400  161306            BIT     #BIT08,@SWR     ;;LOOP ON SPEC. TEST?
      017622  001407                            BEQ     2$              ;;BR IF NO
      017624  017746  161300                    MOV     @SWR,-(SP)      ;;SET DESIRED TEST NUM. FROM SWR
      017630  042716  000340                    BIC     #$SWRMK,(SP)    ;;STRIP AWAY UNDESIRED BITS
      017634  122667  161232                    CMPB    (SP)+,$TSTNM    ;;ON THE RIGHT TEST?
      017640  001465                            BEQ     $OVER           ;;BR IF YES
      017642  105767  161225            2$:     TSTB    $ERFLG          ;;HAS AN ERROR OCCURRED?
      017646  001421                            BEQ     3$              ;;BR IF NO
      017650  126767  161231  161215            CMPB    $ERMAX,$ERFLG   ;;MAX. ERRORS FOR THIS TEST OCCURRED?
```

```
        017656  101015                          BHI     3$                      ;;BR IF NO
        017660  032777  001000  161242          BIT     #BIT09,@SWR             ;;LOOP ON ERROR?
        017666  001404                          BEQ     4$                      ;;BR IF NO
        017670  016767  161204  161200  7$:     MOV     $LPERR,$LPADR           ;;SET LOOP ADDRESS TO LAST SCOPE
        017676  000446                          BR      $OVER
        017700  105067  161167          4$:     CLRB    $ERFLG                  ;;ZERO THE ERROR FLAG
        017704  005067  161250                  CLR     $TIMES                  ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
        017710  000415                          BR      1$                      ;;ESCAPE TO THE NEXT TEST
        017712  032777  004000  161210  3$:     BIT     #BIT11,@SWR             ;;INHIBIT ITERATIONS?
        017720  001011                          BNE     1$                      ;;BR IF YES
        017722  005767  161254                  TST     $PASS                   ;;IF FIRST PASS OF PROGRAM
        017726  001406                          BEQ     1$                      ;;      INHIBIT ITERATIONS
        017730  005267  161140                  INC     $ICNT                   ;;INCREMENT ITERATION COUNT
        017734  026767  001220  161132          CMP     $TIMES,$ICNT            ;;CHECK THE NUMBER OF ITERATIONS MADE
        017742  002024                          BGE     $OVER                   ;;BR IF MORE ITERATION REQUIRED
        017744  012767  000001  161122  1$:     MOV     #1,$ICNT                ;;REINITIALIZE THE ITERATION COUNTER
        017752  016767  000052  161200          MOV     $MXCNT,$TIMES           ;;SET NUMBER OF ITERATIONS TO DO
        017760  105267  161106          $SVLAD: INCB    $TSTNM                  ;;COUNT TEST NUMBERS
        017764  116767  161102  161206          MOVB    $TSTNM,$TESTN           ;;SET TEST NUMBER IN APT MAILBOX
        017772  011667  161100                  MOV     (SP),$LPADR             ;;SAVE SCOPE LOOP ADDRESS
        017776  011667  161076                  MOV     (SP),$LPERR             ;;SAVE ERROR LOOP ADDRESS
        020002  005067  161154                  CLR     $ESCAPE                 ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
        020006  112767  000001  161071          MOVB    #1,$ERMAX               ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
        020014  016777  161052  161110  $OVER:  MOV     $TSTNM,@DISPLAY         ;;DISPLAY TEST NUMBER
        020022  016716  161050                  MOV     $LPADR,(SP)             ;;FUDGE RETURN ADDRESS
        020026  000207                  ENDINS: RTS     PC                      ;EXIT SCOPE ROUTINE BACK TO TEST.
        020030  000001                  $MXCNT: 1                               ;;MAX. NUMBER OF ITERATIONS
3597                                    ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $CKSWR ROUTINE
                                        ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **S.SMAC**.
        020032  106746                          MFPS    -(SP)                   ;PUT THE PROCESSOR STATUS ON THE STACK
        020034  105066  000001                  CLRB    1(SP)                   ;HIGH BYTE CLEARED TO INSURE KERNEL MODE
                                                                                ;ON PSW RETURN.
        020040  004767  000542                  JSR     PC,     $CKSWR          ;GO TO THE SUBROUTINE
3598                                    .SBTTL  ERROR HANDLER ROUTINE
                                        ;;**************************************************************
                                        ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
                                        ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
                                        ;*AND GO TO $ERRTYP ON ERROR
                                        ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
                                        ;*SW15=1         HALT ON ERROR
                                        ;*SW13=1         INHIBIT ERROR TYPEOUTS
                                        ;*SW10=1         BELL ON ERROR
                                        ;*SW09=1         LOOP ON ERROR
                                        ;*CALL
                                        ;*      ERROR   N       ;;ERROR=EMT AND N=ERROR ITEM NUMBER
        020044                          $ERROR:
                                        ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $CKSWR ROUTINE
                                        ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
        020044  106746                          MFPS    -(SP)                   ;PUT THE PROCESSOR STATUS ON THE STACK
        020046  105066  000001                  CLRB    1(SP)                   ;HIGH BYTE CLEARED TO INSURE KERNEL MODE
                                                                                ;ON PSW RETURN.
        020052  004767  000530                  JSR     PC,     $CKSWR          ;GO TO THE SUBROUTINE
        020056  062716  000002                  ADD     #2,     (SP)            ;ADJUST POINTER PAST CODE WORD.
        020062  105267  161005          7$:     INCB    $ERRFLG                 ;;SET THE ERROR FLAG
        020066  001775                          BEQ     7$                      ;;DON'T LET THE FLAG GO TO ZERO
        020070  016777  160776  161034          MOV     $TSTNM,@DISPLAY         ;;DISPLAY TEST NUMBER AND ERROR FLAG
        020076  032777  002000  161024          BIT     #BIT10,@SWR             ;;BELL ON ERROR?
```

```
020104  001403                          BEQ     1$              ;;NO - SKIP
020106  004567  001640                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
020112  001164                          .WORD   $BELL           ;ADDRESS OF MESSAGE TO BE TYPED
020114  005267  160762          1$:     INC     $ERTTL          ;;COUNT THE NUMBER OF ERRORS
020120  011667  160762                  MOV     (SP),$ERRPC     ;;GET ADDRESS OF ERROR INSTRUCTION
020124  162767  000002  160754          SUB     #2,$ERRPC
020132  117767  160750  160744          MOVB    @$ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
020140  032777  020000  160762          BIT     #BIT13,@SWR     ;;SKIP TYPEOUT IF SET
020146  001005                          BNE     20$             ;;SKIP TYPEOUTS
020150  004767  000120                  JSR     PC,$ERRTYP      ;;GO TO USER ERROR ROUTINE
020154  004567  001572                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
020160  001171                          .WORD   $CRLF           ;ADDRESS OF MESSAGE TO BE TYPED
020162                          20$:
020162  122767  000001  161024          CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
020170  001007                          BNE     2$              ;;NO,SKIP APT ERROR REPORT
020172  116767  160706  000004          MOVB    $ITEMB,21$      ;;SET ITEM NUMBER AS ERROR NUMBER
020200  004767  002152                  JSR     PC,$ATY4        ;;REPORT FATAL ERROR TO APT
020204     000          21$:    .BYTE   0
020205     000                  .BYTE   0
020206  000777          22$:    BR      22$             ;;APT ERROR LOOP
020210  005777  160714          2$:     TST     @SWR            ;;HALT ON ERROR
020214  100006                          BPL     3$              ;;SKIP IF CONTINUE
020216  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**.
020220  106746                          MFPS    -(SP)           ;PUT THE PROCESSOR STATUS ON THE STACK
020222  105066  000001                  CLRB    1(SP)           ;HIGH BYTE CLEARED TO INSURE KERNEL MODE
                                                                ;ON PSW RETURN.
020226  004767  000354                  JSR     PC,     $CKSWR  ;GO TO THE SUBROUTINE
020232  032777  001000  160670  3$:     BIT     #BIT09,@SWR     ;;LOOP ON ERROR SWITCH SET?
020240  001402                          BEQ     4$              ;;BR IF NO
020242  016716  160632                  MOV     $LPERR,(SP)     ;;FUDGE RETURN FOR LOOPING
020246  005767  160710          4$:     TST     $ESCAPE         ;;CHECK FOR AN ESCAPE ADDRESS
020252  001402                          BEQ     5$              ;;BR IF NONE
020254  016716  160702                  MOV     $ESCAPE,(SP)    ;;FUDGE RETURN ADDRESS FOR ESCAPE
020260                          5$:
020260  022737  012722  000042          CMP     #$ENDAD,@#42    ;;ACT-11 AUTO-ACCEPT?
020266  001001                          BNE     6$              ;;BRANCH IF NO
020270  000000                          HALT                    ;;YES
020272                          6$:
020272  000207                          RTS     PC
3599                            ;;**************************************************************

                        .SBTTL  ERROR MESSAGE TYPEOUT ROUTINE


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

020274                  $ERRTYP:
020274  004567  001452          JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
020300  001171                  .WORD   $CRLF           ;ADDRESS OF MESSAGE TO BE TYPED
020302  010046                  MOV     R0,-(SP)                ;SAVE R0
020304  005000                  CLR     R0                      ;PICKUP THE ITEM INDEX
020306  156700  160572          BISB    $ITEMB,R0
```

```
020312  001010                          BNE     1$                      ;IF ITEM NUMBER IS ZERO, JUST
                                                                        ;TYPE THE PC OF THE ERROR
020314  016746  160566                  MOV     $ERRPC,-(SP)    ;;SAVE $ERRPC FOR TYPEOUT
                                                                ;;ERROR ADDRESS
                                ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOC ROUTINE
                                ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
020320  106746                          MFPS    -(SP)                   ;PUT THE PROCESSOR STATUS ON THE STACK
020322  105066  000001                  CLRB    1(SP)                   ;HIGH BYTE CLEARED TO INSURE KERNEL MODE
                                                                        ;ON PSW RETURN.
020326  004767  002672                  JSR     PC,     $TYPOC  ;GO TO THE SUBROUTINE
020332  000516                          BR      10$                     ;GET OUT
020334  016767  160546  161142  1$:     MOV     $ERRPC, $VERPC  ;SET UP VIRTUAL PC FOR TYPEOUT.
020342  166767  160164  161134          SUB     RELOCF, $VERPC  ;MAKE VIRTUAL IF NOT ALREADY.
020350  005300                          DEC     R0                      ;ADJUST THE INDEX SO THAT IT WILL
020352  006300                          ASL     R0                      ;       WORK FOR THE ERROR TABLE
020354  006300                          ASL     R0
020356  006300                          ASL     R0
020360  066700  161450                  ADD     .ERRTB, R0      ;FORM TABLE POINTER
020364  012067  000006                  MOV     (R0)+,2$                ;PICKUP "ERROR MESSAGE" POINTER
020370  001406                          BEQ     3$                      ;SKIP TYPEOUT IF NO POINTER
020372  004567  001354                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
020376  000000                  2$:     .WORD   0                       ;"ERROR MESSAGE" POINTER GOES HERE
020400  004567  001346                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
020404  001171                          .WORD   $CRLF                   ;ADDRESS OF MESSAGE TO BE TYPED
020406  012067  000006          3$:     MOV     (R0)+,4$                ;PICKUP "DATA HEADER" POINTER
020412  001406                          BEQ     5$                      ;SKIP TYPEOUT IF 0
020414  004567  001332                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
020420  000000                  4$:     .WORD   0                       ;"DATA HEADER" POINTER GOES HERE
020422  004567  001324                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
020426  001171                          .WORD   $CRLF                   ;ADDRESS OF MESSAGE TO BE TYPED
020430  010146                  5$:     MOV     R1,-(SP)                ;SAVE R1
020432  012001                          MOV     (R0)+,R1                ;PICKUP "DATA TABLE" POINTER
020434  001454                          BEQ     9$                      ;BR IF NO DATA TO BE TYPED
020436  066701  160070                  ADD     RELOCF, R1      ;ADJUST POINTER
020442  012000                          MOV     (R0)+,R0                ;PICKUP "DATA FORMAT" POINTER
020444  066700  160062                  ADD     RELOCF, R0      ;ADJUST POINTER.
020450  105720                  6$:     TSTB    (R0)+                   ;CHECK THE FORMAT
020452  001007                          BNE     7$                      ;BR IF NOT 16-BIT OCTAL
020454  013146                          MOV     @(R1)+,-(SP)    ;;SAVE @(R1)+ FOR TYPEOUT
                                ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOC ROUTINE
                                ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
020456  106746                          MFPS    -(SP)                   ;PUT THE PROCESSOR STATUS ON THE STACK
020460  105066  000001                  CLRB    1(SP)                   ;HIGH BYTE CLEARED TO INSURE KERNEL MODE
                                                                        ;ON PSW RETURN.
020464  004767  002534                  JSR     PC,     $TYPOC  ;GO TO THE SUBROUTINE
020470  000430                          BR      8$
020472  100407                  7$:     BMI     17$                     ;BRANCH IF NOT DECIMAL
020474  013146                          MOV     @(R1)+,-(SP)    ;;SAVE @(R1)+ FOR TYPEOUT
                                ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPDS ROUTINE
                                ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
020476  106746                          MFPS    -(SP)                   ;PUT THE PROCESSOR STATUS ON THE STACK
020500  105066  000001                  CLRB    1(SP)                   ;HIGH BYTE CLEARED TO INSURE KERNEL MODE
                                                                        ;ON PSW RETURN.
020504  004767  002236                  JSR     PC,     $TYPDS  ;GO TO THE SUBROUTINE
020510  000420                          BR      8$                      ;SKIP
020512  122760  177777  177777  17$:     CMPB    # 1,     -1(R0)  ;CHECK FOR 22-BIT ADDRESS FORMAT.
020520  001004                          BNE     18$                     ;BR IF NOT 22-BIT ADDRESS FORMAT.
```

D8


```
020522  013146                          MOV     @(R1)+,  (SP)   ;PUT THE DATA ON THE STACK.
020524  004767  002736                   JSR     PC,      $TYPAD ;DETERMINE THE PHYSICAL ADDRESS AND TYPE IT.
020530  000410                          BR      8$              ;SKIP
020532                          18$:
020532  005046                          CLR     -(SP)           ;CLEAR THE WORD ON THE STACK.
020534  113116                          MOVB    @(R1)+, (SP)    ;PUT THE DATA ON THE STACK.
                                ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOS ROUTINE
                                ;*  WIHTOUT USING A "TRAP' INSTRUCTION AS CALLED FOR BY **SYSMAC**.
020536  106746                          MFPS    -(SP)           ;PUT THE PROCESSOR STATUS ON THE STACK
020540  105066  000001                   CLRB    1(SP)           ;HIGH BYTE CLEARED TO INSURE KERNEL MODE
                                                                ;ON PSW RETURN.
020544  004767  002430                   JSR     PC,      $TYPOS ;GO TO THE SUBROUTINE
020550     003                          .BYTE   3               ;TYPE 3 DIGITS.
020551     001                          .BYTE   1               ;TYPE LEADING ZEROS.
020552  005711                  8$:     TST     (R1)                    ;IS THERE ANOTHER NUMBER?
020554  001404                          BEQ     9$                      ;BR IF NO
020556  004567  001170                   JSR     R5,      $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
020562  020602                          .WORD   11$             ;ADDRESS OF MESSAGE TO BE TYPED
020564  000731                          BR      6$                      ;LOOP

020566  012601                  9$:     MOV     (SP)+,R1                ;RESTORE R1
020570  012600                  10$:    MOV     (SP)+,R0                ;RESTORE R0
020572  004567  001154                   JSR     R5,      $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
020576  001171                          .WORD   $CRLF           ;ADDRESS OF MESSAGE TO BE TYPED
020600  000207                          RTS     PC              ;RETURN
020602     040     040     000  11$:    .ASCIZ  /  /            ;DOUBLE SPACE
                                        .EVEN
3600                                    .SBTTL  TTY INPUT ROUTINE
                                ;;**************************************************************
                                        .ENABL  LSB
                                ;;**************************************************************
                                ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
                                ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
                                ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
                                ;*WHEN OPERATING IN TTY FLAG MODE.
020606  022767  000176  160314  $CKSWR: CMP     #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED?
020614  001105                          BNE     15$             ;;BRANCH IF NO
020616  105777  160312                   TSTB    @$TKS           ;;CHAR THERE?
020622  100102                          BPL     15$             ;;IF NO, DON'T WAIT AROUND
020624  117746  160306                   MOVB    @$TKB,-(SP)     ;;SAVE THE CHAR
020630  042716  177600                   BIC     #^C177,(SP)     ;;STRIP-OFF THE ASCII
020634  022726  000007                   CMP     #7,(SP)+        ;;IS IT A CONTROL G?
020640  001073                          BNE     15$             ;;NO, RETURN TO USER
020642  126727  160256  000001           CMPB    $AUTOB,#1       ;;ARE WE RUNNING IN AUTO-MODE?
020650  001467                          BEQ     15$             ;;BRANCH IF YES
020652  004567  001074                   JSR     R5,      $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
020656  021547                          .WORD   $CNTLG          ;ADDRESS OF MESSAGE TO BE TYPED
020660                          $GTSWR:
020660  004567  001066                   JSR     R5,      $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
020664  021554                          .WORD   $MSWR           ;ADDRESS OF MESSAGE TO BE TYPED
020666  016746  157304                   MOV     SWREG,-(SP)     ;;SAVE SWREG FOR TYPEOUT
                                ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOC ROUTINE
                                ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
020672  106746                          MFPS    -(SP)           ;PUT THE PROCESSOR STATUS ON THE STACK
020674  105066  000001                   CLRB    1(SP)           ;HIGH BYTE CLEARED TO INSURE KERNEL MODE
                                                                ;ON PSW RETURN.
020700  004767  002320                   JSR     PC,      $TYPOC ;GO TO THE SUBROUTINE
```

```
020704  004567  001042                    JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
020710  021565                            .WORD   $MNEW           ;ADDRESS OF MESSAGE TO BE TYPED
020712  005046                  19$:      CLR      (SP)           ;;CLEAR COUNTER
020714  005046                            CLR     -(SP)           ;;THE NEW SWR
020716  105777  160212          7$:       TSTB    @$TKS           ;;CHAR THERE?
020722  100375                            BPL     7$              ;;IF NOT TRY AGAIN
020724  117746  160206                    MOVB    @$TKB, (SP)     ;;PICK UP CHAR
020730  042716  177600                    BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
020734  021627  000025          9$:       CMP     (SP),#25        ;;IS IT A CONTROL-U?
020740  001006                            BNE     10$             ;;BRANCH IF NOT
020742  004567  001004                    JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
020746  021542                            .WORD   $CNTLU          ;ADDRESS OF MESSAGE TO BE TYPED
020750  062706  000006          20$:      ADD     #6,SP           ;;IGNORE PREVIOUS INPUT
020754  000756                            BR      19$             ;;LET'S TRY IT AGAIN
020756  021627  000015          10$:      CMP     (SP),#15        ;;IS IT A <CR>?
020762  001023                            BNE     16$             ;;BRANCH IF NO
020764  005766  000004                    TST     4(SP)           ;;YES, IS IT THE FIRST CHAR?
020770  001403                            BEQ     11$             ;;BRANCH IF YES
020772  016677  000002  160130            MOV     2(SP),@SWR      ;;SAVE NEW SWR
021000  062706  000006          11$:      ADD     #6,SP           ;;CLEAR UP STACK
021004                          14$:
021004  004567  000742                    JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
021010  001171                            .WORD   $CRLF           ;ADDRESS OF MESSAGE TO BE TYPED
021012  126727  160107  000001            CMPB    $INTAG,#1       ;;RE-ENABLE TTY KBD INTERRUPTS?
021020  001003                            BNE     15$             ;;BRANCH IF NOT
021022  012777  000100  160104            MOV     #100,@$TKS      ;;RE-ENABLE TTY KBD INTERRUPTS
021030  000002                  15$:      RTI                     ;;RETURN
021032  004767  001160          16$:      JSR     PC,$TYPEC       ;;ECHO CHAR
021036  021627  000060                    CMP     (SP),#60        ;;CHAR < 0?
021042  002420                            BLT     18$             ;;BRANCH IF YES
021044  021627  000067                    CMP     (SP),#67        ;;CHAR > 7?
021050  003015                            BGT     18$             ;;BRANCH IF YES
021052  042726  000060                    BIC     #60,(SP)+       ;;STRIP-OFF ASCII
021056  005766  000002                    TST     2(SP)           ;;IS THIS THE FIRST CHAR
021062  001403                            BEQ     17$             ;;BRANCH IF YES
021064  006316                            ASL     (SP)            ;;NO, SHIFT PRESENT
021066  006316                            ASL     (SP)            ;;   CHAR OVER TO MAKE
021070  006316                            ASL     (SP)            ;;   ROOM FOR NEW ONE.
021072  005266  000002          17$:      INC     2(SP)           ;;KEEP COUNT OF CHAR
021076  056616  177776                    BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
021102  000705                            BR      7$              ;;GET THE NEXT ONE
021104                          18$:
021104  004567  000642                    JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
021110  001170                            .WORD   $QUES           ;ADDRESS OF MESSAGE TO BE TYPED
021112  000716                            BR      20$             ;;SIMULATE CONTROL-U
                                .DSABL  LSB
                                ;;*****************************************************************
                                ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
                                ;*CALL:
                                ;*      RDCHR                   ;;INPUT A SINGLE CHARACTER FROM THE TTY
                                ;*      RETURN HERE             ;;CHARACTER IS ON THE STACK
                                ;*                              ;;WITH PARITY BIT STRIPPED OFF
                                ;
021114  011646                  $RDCHR: MOV     (SP), (SP)      ;;PUSH DOWN THE PC
021116  016666  000004  000002          MOV     4(SP),2(SP)     ;;SAVE THE PS
021124  105777  160004          1$:     TSTB    @$TKS           ;;WAIT FOR
021130  100375                          BPL     1$              ;;A CHARACTER
```

F8


```
021132  117766  160000  000004          MOVB    @$TKB,4(SP)     ;;READ THE TTY
021140  042766  177600  000004          BIC     #^C<177>,4(SP)  ;;GET RID OF JUNK IF ANY
021146  026627  000004  000023          CMP     4(SP),#23       ;;IS IT A CONTROL-S?
021154  001013                          BNE     3$              ;;BRANCH IF NO
021156  105777  157752          2$:     TSTB    @$TKS           ;;WAIT FOR A CHARACTER
021162  100375                          BPL     2$              ;;LOOP UNTIL ITS THERE
021164  117746  157746                  MOVB    @$TKB,-(SP)     ;;GET CHARACTER
021170  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
021174  022627  000021                  CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
021200  001366                          BNE     2$              ;;IF NOT DISCARD IT
021202  000750                          BR      1$              ;;YES, RESUME
021204  026627  000004  000021  3$:     CMP     4(SP),#$XON     ;;IS IT A RANDOM XON?           ;RAN001
021212  001744                          BEQ     1$              ;;BRANCH IF YES                 ;RAN001
021214  026627  000004  000140          CMP     4(SP),#140      ;;IS IT UPPER CASE?
021222  002407                          BLT     4$              ;;BRANCH IF YES
021224  026627  000004  000175          CMP     4(SP),#175      ;;IS IT A SPECIAL CHAR?
021232  003003                          BGT     4$              ;;BRANCH IF YES
021234  042766  000040  000004          BIC     #40,4(SP)       ;;MAKE IT UPPER CASE
021242  000002                  4$:     RTI                     ;;GO BACK TO USER
                        ;;**************************************************************
                        ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
                        ;*CALL:
                        ;*      RDLIN                           ;;INPUT A STRING FROM THE TTY
                        ;*      RETURN HERE                     ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
                        ;*                                      ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
021244  010346                 $RDLIN. MOV     R3,-(SP)        ;;SAVE R3
021246  005046                         CLR     -(SP)           ;;CLEAR THE RUBOUT KEY
021250  012703  021532          1$:     MOV     #$TTYIN,R3      ;;GET ADDRESS
021254  022703  021542          2$:     CMP     #$TTYIN+8.,R3   ;;BUFFER FULL?
021260  101470                          BLOS    4$              ;;BR IF YES
                        ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $RDCHR ROUTINE
                        ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
021262  106746                          MFPS    -(SP)           ;PUT THE PROCESSOR STATUS ON THE STACK
021264  105066  000001                  CLRB    1(SP)           ;HIGH BYTE CLEARED TO INSURE KERNEL MODE
                                                                ;ON PSW RETURN.
021270  004767  177620                  JSR     PC,     $RDCHR  ;GO TO THE SUBROUTINE
021274  112613                          MOVB    (SP)+,(R3)      ;;GET CHARACTER
021276  122713  000177          10$:    CMPB    #177,(R3)       ;;IS IT A RUBOUT
021302  001024                          BNE     5$              ;;BR IF NO
021304  005716                          TST     (SP)            ;;IS THIS THE FIRST RUBOUT?
021306  001010                          BNE     6$              ;;BR IF NO
021310  112767  000134  000212          MOVB    #'\,9$          ;;TYPE A BACK SLASH
021316  004567  000430                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
021322  021530                          .WORD   9$              ;ADDRESS OF MESSAGE TO BE TYPED
021324  012716  177777                  MOV     #-1,(SP)        ;;SET THE RUBOUT KEY
021330  005303                  6$:     DEC     R3              ;;BACKUP BY ONE
021332  020327  021532                  CMP     R3,#$TTYIN      ;;STACK EMPTY?
021336  103441                          BLO     4$              ;;BR IF YES
021340  111367  000164                  MOVB    (R3),9$         ;;SETUP TO TYPEOUT THE DELETED CHAR.
021344  004567  000402                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
021350  021530                          .WORD   9$              ;ADDRESS OF MESSAGE TO BE TYPED
021352  000740                          BR      2$              ;;GO READ ANOTHER CHAR.
021354  005716                  5$:     TST     (SP)            ;;RUBOUT KEY SET?
021356  001407                          BEQ     7$              ;;BR IF NO
021360  112767  000134  000142          MOVB    #'\,9$          ;;TYPE A BACK SLASH
021366  004567  000360                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
021372  021530                          .WORD   9$              ;ADDRESS OF MESSAGE TO BE TYPED
```

```
021374  005016                          CLR     (SP)            ;;CLEAR THE RUBOUT KEY
021376  122713  000025          7$:     CMPB    #25,(R3)        ;;IS CHARACTER A CTRL U?
021402  001004                          BNE     8$              ;;BR IF NO
021404  004567  000342                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
021410  021542                          .WORD   $CNTLU          ;ADDRESS OF MESSAGE TO BE TYPED
021412  000716                          BR      1$              ;;GO START OVER
021414  122713  000022          8$:     CMPB    #22,(R3)        ;;IS CHARACTER A 'tR'?
021420  001014                          BNE     3$              ;;BRANCH IF NO
021422  105013                          CLRB    (R3)            ;;CLEAR THE CHARACTER
021424  004567  000322                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
021430  001171                          .WORD   $CRLF           ;ADDRESS OF MESSAGE TO BE TYPED
021432  004567  000314                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
021436  021532                          .WORD   $TTYIN          ;ADDRESS OF MESSAGE TO BE TYPED
021440  000705                          BR      2$              ;;GO PICKUP ANOTHER CHACTER
021442                          4$:
021442  004567  000304                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
021446  001170                          .WORD   $QUES           ;ADDRESS OF MESSAGE TO BE TYPED
021450  000677                          BR      1$              ;;CLEAR THE BUFFER AND LOOP
021452  111367  000052          3$:     MOVB    (R3),9$         ;;ECHO THE CHARACTER
021456  004567  000270                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
021462  021530                          .WORD   9$              ;ADDRESS OF MESSAGE TO BE TYPED
021464  122723  000015                  CMPB    #15,(R3)+       ;;CHECK FOR RETURN
021470  001271                          BNE     2$              ;;LOOP IF NOT RETURN
021472  105063  177777                  CLRB    -1(R3)          ;;CLEAR RETURN (THE 15)
021476  004567  000250                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
021502  001172                          .WORD   $LF             ;ADDRESS OF MESSAGE TO BE TYPED
021504  005726                          TST     (SP)+           ;;CLEAN RUBOUT KEY FROM THE STACK
021506  012603                          MOV     (SP)+,R3        ;;RESTORE R3
021510  011646                          MOV     (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
021512  016666  000004  000002          MOV     4(SP),2(SP)     ;;      FIRST ASCII CHARACTER ON IT
021520  012766  021532  000004          MOV     #$TTYIN,4(SP)
021526  000002                          RTI                     ;;RETURN
021530     000                  9$:     .BYTE   0               ;;STORAGE FOR ASCII CHAR. TO TYPE
021531     000                          .BYTE   0               ;;TERMINATOR
021532                          $TTYIN: .BLKB   8.              ;;RESERVE 8 BYTES FOR TTY INPUT
021542     136     125     015  $CNTLU: .ASCIZ  /tU/<15><12>    ;;CONTROL "U"
021545     012     000
021547     136     107     015  $CNTLG: .ASCIZ  /tG/<15><12>    ;;CONTROL "G"
021552     012     000
021554     015     012     123  $MSWR:  .ASCIZ  <15><12>/SWR = /
021557     127     122     040
021562     075     040     000
021565     040     040     116  $MNEW:  .ASCIZ  /  NEW = /
021570     105     127     040
021573     075     040     000
3601                                    .SBTTL  READ AN OCTAL NUMBER FROM THE TTY
                                ;;**************************************************************
                                ;*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
                                ;*CHANGE IT TO BINARY.
                                ;*THE INPUT CHARACTERS WILL BE CHECKED TO INSURED THEY ARE LEGAL
                                ;*OCTAL DIGITS. IF AN ILLEGAL CHARACTER IS READ A "?" WILL BE TYPED
                                ;*FOLLOWED BY A CARRIAGE RETURN-LINE FEED. THE COMPLETE NUMBER MUST
                                ;*THEN BE RETYPED. THE INPUT IS TERMINATED BY TYPING A CARRIAGE RETURN.
                                ;*CALL:
                                ;*      RDOCT                   ;;READ AN OCTAL NUMBER
                                ;*      RETURN HERE             ;;LOW ORDER BITS ARE ON TOP OF THE STACK
                                ;*                              ;;HIGH ORDER BITS ARE IN $HIOCT
```

```
021576  011646                  $RDOCT: MOV     (SP),-(SP)      ;;PROVIDE SPACE FOR THE
021600  016666  000004  000002          MOV     4(SP),2(SP)     ;;INPUT NUMBER
021606  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
021610  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
021612  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
021614                          1$:
                                ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $RDLIN ROUTINE
                                ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
021614  106746                          MFPS    -(SP)           ;PUT THE PROCESSOR STATUS ON THE STACK
021615  105066  000001                  CLRB    1(SP)           ;HIGH BYTE CLEARED TO INSURE KERNEL MODE
                                                                ;ON PSW RETURN.
021622  004767  177416                  JSR     PC,     $RDLIN  ;GO TO THE SUBROUTINE
021626  012600                          MOV     (SP)+,R0        ;;GET ADDRESS OF 1ST CHARACTER
021630  010067  000102                  MOV     R0,5$           ;;AND SAVE IT
021634  005001                          CLR     R1              ;;CLEAR DATA WORD
021636  005002                          CLR     R2
021640  112046                  2$:     MOVB    (R0)+, (SP)     ;;PICKUP THIS CHARACTER
021642  001420                          BEQ     3$              ;;IF ZERO GET OUT
021644  122716  000060                  CMPB    #'0,(SP)        ;;MAKE SURE THIS CHARACTER
021650  003026                          BGT     4$              ;;IS AN OCTAL DIGIT
021652  122716  000067                  CMPB    #'7,(SP)
021656  002423                          BLT     4$
021660  006301                          ASL     R1              ;;*2
021662  006102                          ROL     R2
021664  006301                          ASL     R1              ;;*4
021666  006102                          ROL     R2
021670  006301                          ASL     R1              ;;*8
021672  006102                          ROL     R2
021674  042716  177770                  BIC     #^C7,(SP)       ;;STRIP THE ASCII JUNK
021700  062601                          ADD     (SP)+,R1        ;;ADD IN THIS DIGIT
021702  000756                          BR      2$              ;;LOOP
021704  005726                  3$:     TST     (SP)+           ;;CLEAN TERMINATOR FROM STACK
021706  010166  000012                  MOV     R1,12(SP)       ;;SAVE THE RESULT
021712  010267  000032                  MOV     R2,$HIOCT
021716  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
021720  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
021722  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
021724  000002                          RTI                     ;;RETURN
021726  005726                  4$:     TST     (SP)+           ;;CLEAN PARTIAL FROM STACK
021730  105010                          CLRB    (R0)            ;;SET A TERMINATOR
021732  004567  000014                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
021736  000000                  5$:     .WORD   0
021740  004567  000006                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
021744  001170                          .WORD   $QUES           ;ADDRESS OF MESSAGE TO BE TYPED
021746  000722                          BR      1$              ;;TRY AGAIN
021750  000000                  $HIOCT: .WORD   0               ;;HIGH ORDER BITS GO HERE
3602
3603                                    ;;*****************************************************************
3604                                    ;* SUBROUTINE TO PASS RELOCATED MESSAGE ADDRESSES TO THE $TYPE ROUTINE.
3605                                    ;*  CALL:       JSR     R5,     $PRINT
3606                                    ;*              <MESSAGE VIRTUAL ADDRESS>
3607                                    ;;*****************************************************************
3608 021752  012567  000020      $PRINT: MOV     (R5)+,  1$      ;GET THE MESSAGE VIRTUAL ADDRESS.
3609 021756  066767  156550  000012      ADD     RELOC,  1$      ;MAKE IT PHYSICAL.
3610                                    ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPE ROUTINE
                                        ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
021764  106746                          MFPS    -(SP)           ;PUT THE PROCESSOR STATUS ON THE STAC<
```

```
        021766  105066  000001                  CLRB    1(SP)           ;HIGH BYTE CLEARED TO INSURE KERNEL MODE
                                                                        ;ON PSW RETURN.
        021772  004767  0C0004                  JSR     PC,     $TYPE   ;GO TO THE SUBROUTINE
  3611  021776  000000                  1$:     .WORD   0               ;CONTAINS THE PHYSICAL MESSAGE ADDRESS.
  3612  022000  000205                          RTS     R5              ;RETURN.
  3613
  3614                                  .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
                                        ;*
        022002  105767  157141          $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
        022006  100002                          BPL     1$              ;;BR IF YES
        022010  000000                          HALT                    ;;HALT HERE IF NO TERMINAL
        022012  000430                          BR      3$              ;;LEAVE
        022014  010046                  1$:     MOV     R0,-(SP)        ;;SAVE R0
        022016  017600  000002                  MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
        022022  122767  000001  157164          CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
        022030  001011                          BNE     62$             ;;NO,GO CHECK FOR APT CONSOLE
        022032  132767  000100  157155          BITB    #APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
        022040  001405                          BEQ     62$             ;;NO,GO CHECK FOR CONSOLE
        022042  010067  000004                  MOV     R0,61$          ;;SETUP MESSAGE ADDRESS FOR APT
        022046  004767  000274                  JSR     PC,$ATY3        ;;SPOOL MESSAGE TO APT
        022052  000000                  61$:    .WORD   0               ;;MESSAGE ADDRESS
        022054  132767  000040  157133  62$:    BITB    #APTCSUP,$ENVM  ;;APT CONSOLE SUPPRESSED
        022062  001003                          BNE     60$             ;;YES,SKIP TYPE OUT
        022064  112046                  2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
        022066  001005                          BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
        022070  005726                          TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STAC<
        022072  012600                  60$:    MOV     (SP)+,R0        ;;RESTORE R0
        022074  062716  000002          3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
        022100  000002                          RTI                     ;;RETURN
        022102  122716  000011          4$:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
        022106  001431                          BEQ     8$
        022110  122716  000200                  CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
        022114  001007                          BNE     5$
        022116  005726                          TST     (SP)+           ;;POP  <CR><LF> EQUIV
        022120  004567  177626                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
        022124  001171                          $CRLF
        022126  105067  000202                  CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
        022132  000754                          BR      2$              ;;GET NEXT CHARACTER
        022134  004767  000056          5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
        022140  126726  157002          6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
        022144  001347                          BNE     2$              ;;IF NO GO GET NEXT CHAR.
        022146  016746  156772                  MOV     $NULL, (SP)     ;;GET # OF FILLER CHARS. NEEDED
                                                                        ;;AND THE NULL CHAR.
        022152  105366  000001          7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
```

```
022156  002770                  BLT     6$              ;;BR IF NO -GO POP THE NULL OFF OF STAC<
022160  004767  000032          JSR     PC,$TYPEC       ;;GO TYPE A NULL
022164  105367  000144          DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
022170  000770                  BR      7$              ;;LOOP
                        ;HORIZONTAL TAB PROCESSOR
022172  112716  000040  8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
022176  004767  000014  9$:     JSR     PC,$TYPEC       ;;TYPE A SPACE
022202  132767  000007  000124  BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
022210  001372                  BNE     9$              ;;TAB STOP
022212  005726                  TST     (SP)+           ;;POP SPACE OFF STACK
022214  000723                  BR      2$              ;;GET NEXT CHARACTER
022216                  $TYPEC:
022216  105777  156712          TSTB    @$TKS           ;;CHAR IN KYBD BUFFER?          ;MJD001
022222  100022                  BPL     10$             ;;BR IF NOT                     ;MJD001
022224  017746  156706          MOV     @$TKB,-(SP)     ;;GET CHAR                      ;MJD001
022230  042716  177600          BIC     #177600,(SP)    ;;STRIP EXTRANEOUS BITS         ;MJD001
022234  122716  000023          CMPB    #$XOFF,(SP)     ;;WAS CHAR XOFF                 ;MJD001
022240  001012                  BNE     102$            ;;BR IF NOT                     ;MJD001
022242                  101$:                                                           ;MJD001
022242  105777  156666          TSTB    @$TKS           ;;WAIT FOR CHAR                 ;MJD001
022246  100375                  BPL     101$                                            ;MJD001
022250  117716  156662          MOVB    @$TKB,(SP)      ;;GET CHAR                      ;MJD001
022254  042716  177600          BIC     #177600,(SP)    ;;STRIP IT                      ;MJD001
022260  122716  000021          CMPB    #$XON,(SP)      ;;WAS IT XON?                   ;MJD001
022264  001366                  BNE     101$            ;;BR IF NOT                     ;MJD001
022266                  102$:                                                           ;MJD001
022266  005726                  TST     (SP)+           ;;FIX STACK                     ;MJD001
022270                  10$:                                                            ;MJD001
022270  105777  156644          TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
022274  100375                  BPL     10$                                             ;MJD001
022276  116677  000002  156636  MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
022304  122766  000015  000002  CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
022312  001003                  BNE     1$              ;;BRANCH IF NO
022314  105067  000014          CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
022320  000406                  BR      $TYPEX          ;;EXIT
022322  122766  000012  000002  1$:     CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
022330  001402                  BEQ     $TYPEX          ;;BRANCH IF YES
022332  105227                  INCB    (PC)+           ;;COUNT THE CHARACTER
022334  000000          $CHARCNT:.WORD  0               ;;CHARACTER COUNT STORAGE
022336  000207          $TYPEX: RTS     PC
3615                    .SBTTL  APT COMMUNICATIONS ROUTINE
                        ;;**************************************************************
022340  112767  000001  000376  $ATY1:  MOVB    #1,$FFLG        ;;TO REPORT FATAL ERROR
022346  112767  000001  000366  $ATY3:  MOVB    #1,$MFLG        ;;TO TYPE A MESSAGE
022354  000403                  BR      $ATYC
022356  112767  000001  000360  $ATY4:  MOVB    #1,$FFLG        ;;TO ONLY REPORT FATAL ERROR
022364                  $ATYC:
022364  010046                  MOV     R0,-(SP)        ;;PUSH R0 ON STACK
022366  010146                  MOV     R1,-(SP)        ;;PUSH R1 ON STACK
022370  105767  000346          TSTB    $MFLG           ;;SHOULD TYPE A MESSAGE?
022374  001450                  BEQ     5$              ;;IF NOT:  BR
022376  122767  000001  156610  CMPB    #APTENV,$ENV    ;;OPERATING UNDER APT?
022404  001031                  BNE     3$              ;;IF NOT:  BR
022406  132767  000100  156601  BITB    #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
022414  001425                  BEQ     3$              ;;IF NOT:  BR
022416  017600  000004          MOV     @4(SP),R0       ;;GET MESSAGE ADDR.
022422  062766  000002  000004  ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
```

```
022430  005767  156540          1$:     TST     $MSGTYPE        ;;SEE IF DONE W/ LAST XMISSION?
022434  001375                          BNE     1$              ;;IF NOT:  WAIT
022436  010067  156546                  MOV     R0,$MSGAD       ;;PUT ADDR IN MAILBOX
022442  105720                  2$:     TSTB    (R0)+           ;;FIND END OF MESSAGE
022444  001376                          BNE     2$
022446  166700  156536                  SUB     $MSGAD,R0       ;;SUB START OF MESSAGE
022452  006200                          ASR     R0              ;;GET MESSAGE LNGTH IN WORDS
022454  010067  156532                  MOV     R0,$MSGLGT      ;;PUT LENGTH IN MAILBOX
022460  012767  000004  156506          MOV     #4,$MSGTYPE     ;;TELL APT TO TAKE MSG.
022466  000413                          BR      5$
022470  017667  000004  000016  3$:     MOV     @4(SP),4$       ;;PUT MSG ADDR IN JSR LINKAGE
022476  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDRESS
022504  016746  155266                  MOV     177776,-(SP)    ;;PUSH 177776 ON STACK
022510  004767  177266                  JSR     PC,$TYPE        ;;CALL TYPE MACRO
022514  000000                  4$:     .WORD   0
022516                          5$:
022516  105767  000221                  TSTB    $LFLG           ;;SHOULD LOG AN ERROR?
022522  001422                          BEQ     10$             ;;IF NOT:  BR
022524  017600  000004                  MOV     @4(SP),R0               ;;GET ERROR #
022530  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
022536  012701  001334                  MOV     #$ASTAT,R1      ;;POINT TO TABLE START
022542  005711                  6$:     TST     (R1)            ;;END OF TABLE?
022544  100404                          BMI     8$              ;;IF SO:  BR
022546  020021                          CMP     R0,(R1)+        ;;PROPER ENTRY?
022550  001406                          BEQ     9$              ;;IF SO:  BR
022552  005721                          TST     (R1)+           ;;MOVE PAST COUNTER WORD
022554  000772                          BR      6$              ;;KEEP LOOKING
022556  026701  156720          8$:     CMP     $APTR,R1        ;;TABLE FULL?
022562  001402                          BEQ     10$             ;;IF SO:  BR -- NO MORE ROOM
022564  010021                          MOV     R0,(R1)+        ;;SET UP NEW ENTRY
022566  005211                  9$:     INC     (R1)            ;;BUMP ERROR COUNT
022570  105767  000150          10$:    TSTB    $FFLG           ;;SHOULD REPORT FATAL ERROR?
022574  001416                          BEQ     12$             ;;IF NOT:  BR
022576  005767  156412                  TST     $ENV            ;;RUNNING UNDER APT?
022602  001413                          BEQ     12$             ;;IF NOT:  BR
022604  005767  156364          11$:    TST     $MSGTYPE        ;;FINISHED LAST MESSAGE?
022610  001375                          BNE     11$             ;;IF NOT:  WAIT
022612  017667  000004  156356          MOV     @4(SP),$FATAL   ;;GET ERROR #
022620  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
022626  005267  156342                  INC     $MSGTYPE        ;;TELL APT TO TAKE ERROR
022632  105067  000106          12$:    CLRB    $FFLG           ;;CLEAR FATAL FLAG
022636  105067  000101                  CLRB    $LFLG           ;;CLEAR LOG FLAG
022642  105067  000074                  CLRB    $MFLG           ;;CLEAR MESSAGE FLAG
022646  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
022650  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
022652  000207                          RTS     PC              ;;RETURN
022654                          $ATY6:
022654  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
022656  016700  156620                  MOV     $APTR,R0
022662  162700  001334                  SUB     #$ASTAT,R0      ;;GET SIZE OF STAT TABLE
022666  005767  156302          1$:     TST     $MSGTY          ;;SEE IF DONE LAST COMMUNICATION
022672  001375                          BNE     1$              ;;IF NOT:  WAIT
022674  010067  156312                  MOV     R0,$MSGLG       ;;SET MESSAGE LENGTH
022700  012767  001334  156302          MOV     #$ASTAT,$MSGAD  ;;SET MESSAGE ADDR.
022706  012767  000002  156260          MOV     #2,$MSGTY       ;;TELL APT TO TAKE STATS.
022714  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
022716  000207                          RTS     PC              ;;RETURN
```

```
022720                          $ATY7:
022720  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
022722  012701  001334                  MOV     #$ASTAT,R1      ;;GET START OF TABLE
022726  005721                  1$:     TST     (R1)+           ;;END OF TABLE?
022730  100402                          BMI     2$              ;;IF SO:  BR
022732  005021                          CLR     (R1)+           ;;CLEAR ERROR COUNT
022734  000774                          BR      1$              ;;KEEP CLEARING
022736                          2$:
022736  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
022740  000207                          RTS     PC              ;;RETURN
022742     000                  $MFLG:  .BYTE   0               ;;MESSG. FLAG
022743     000                  $LFLG:  .BYTE   0               ;;LOG FLAG
022744     000                  $FFLG:  .BYTE   0               ;;FATAL FLAG
                                        .EVEN
            000200              APTSIZE=200
            000001              APTENV=001
            000100              APTSPOOL=100
            000040              APTCSUP=040
  3616                          ;;*****************************************************************

                                .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

                                ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
                                ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
                                ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
                                ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
                                ;*REPLACED WITH SPACES.
                                ;*CALL:
                                ;*      MOV     NUM,-(SP)       ;;PUT THE BINARY NUMBER ON THE STACK
                                ;*      TYPDS                   ;;GO TO THE ROUTINE

022746                          $TYPDS:
022746  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
022750  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
022752  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
022754  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
022756  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
022760  012746  020200                  MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
022764  016605  000020                  MOV     20(SP),R5       ;;GET THE INPUT NUMBER
022770  100004                          BPL     1$              ;;BR IF INPUT IS POS.
022772  005405                          NEG     R5              ;;MAKE THE BINARY NUMBER POS.
022774  112766  000055  000001          MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
023002  016700  155524          1$:     MOV     RELOCF, R0      ;GET RELOCATION FACTOR.
023006  012703  023170                  MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
023012  060003                          ADD     R0,     R3      ;ADD IN RELOCATION FACTOR.
023014  112723  000040                  MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
023020  005002                  2$:     CLR     R2              ;;CLEAR THE BCD NUMBER
023022  016001  023160                  MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
023026  160105                  3$:     SUB     R1,R5           ;;FORM THIS BCD DIGIT
023030  002402                          BLT     4$              ;;BR IF DONE
023032  005202                          INC     R2              ;;INCREASE THE BCD DIGIT BY 1
023034  000774                          BR      3$
023036  060105                  4$:     ADD     R1,R5           ;;ADD BACK THE CONSTANT
023040  005702                          TST     R2              ;;CHECK IF BCD DIGIT=0
023042  001002                          BNE     5$              ;;FALL THROUGH IF 0
023044  105716                          TSTB    (SP)            ;;STILL DOING LEADING 0'S?
023046  100407                          BMI     7$              ;;BR IF YES
```

M8


```
023050  106316                  5$:     ASLB    (SP)            ;;MSD?
023052  103003                          BCC     6$              ;;BR IF NO
023054  116663  000001  177777          MOVB    1(SP), 1(R3)    ;;YES--SET THE SIGN
023062  052702  000060          6$:     BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
023066  052702  000040          7$:     BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
023072  110223                          MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
023074  005720                          TST     (R0)+           ;;JUST INCREMENTING
023076  020067  156734                  CMP     R0,     .EIGHT  ;CHECK THE TABLE INDEX
023102  103746                          BLO     2$              ;GO DO THE NEXT DIGIT
023104  101002                          BHI     8$              ;;GO TO EXIT
023106  010502                          MOV     R5,R2           ;;GET THE LSD
023110  000764                          BR      6$              ;;GO CHANGE TO ASCII
023112  105726                  8$:     TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
023114  100003                          BPL     9$              ;;BR IF NO
023116  116063  177777  177776          MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
023124  105013                  9$:     CLRB    (R3)            ;;SET THE TERMINATOR
023126  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
023130  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
023132  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
023134  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
023136  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
023140  004567  176606                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
023144  023170                          .WORD   $DBLK           ;ADDRESS OF MESSAGE TO BE TYPED
023146  016666  000002  000004          MOV     2(SP),4(SP)     ;;ADJUST THE STACK
023154  012616                          MOV     (SP)+,(SP)
023156  000002                          RTI                     ;;RETURN TO USER
023160  023420                  $DTBL:  10000.
023162  001750                          1000.
023164  000144                          100.
023166  000012                          10.
023170                          $DBLK:  .BLKW   4
3617                            .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
                                ;;**************************************************************
                                ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
                                ;*OCTAL (ASCII) NUMBER AND TYPE IT.
                                ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
                                ;*CALL:
                                ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                                ;*      TYPOS                   ;;CALL FOR TYPEOUT
                                ;*      .BYTE   N               ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
                                ;*      .BYTE   M               ;;M=1 OR 0
                                ;*                                      ;;1=TYPE LEADING ZEROS
                                ;*                                      ;;0=SUPPRESS LEADING ZEROS
                                ;*
                                ;*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
                                ;*$TYPOS OR $TYPOC
                                ;*CALL:
                                ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                                ;*      TYPON                   ;;CALL FOR TYPEOUT
                                ;*
                                ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
                                ;*CALL:
                                ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                                ;*      TYPOC                   ;;CALL FOR TYPEOUT
023200  017646  000000          $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
023204  116667  000001  000213          MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
023212  112667  000211                  MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
```

```
023216  062716  000002                  ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
023222  000406                          BR      $TYPON
023224  112767  000001  000173  $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
023232  112767  000006  000167          MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
023240  112767  000005  000156  $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
023246  010346                          MOV     R3,-(SP)        ;;SAVE R3
023250  010446                          MOV     R4,-(SP)        ;;SAVE R4
023252  010546                          MOV     R5,-(SP)        ;;SAVE R5
023254  116704  000147                  MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
023260  005404                          NEG     R4
023262  062704  000006                  ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
023266  110467  000134                  MOVB    R4,$OMODE       ;;SAVE IT FOR USE
023272  116704  000127                  MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
023276  016605  000012                  MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
023302  005003                          CLR     R3              ;;CLEAR THE OUTPUT WORD
023304  006105                  1$:     ROL     R5              ;;ROTATE MSB INTO "C"
023306  000404                          BR      3$              ;;GO DO MSB
023310  006105                  2$:     ROL     R5              ;;FORM THIS DIGIT
023312  006105                          ROL     R5
023314  006105                          ROL     R5
023316  010503                          MOV     R5,R3
023320  006103                  3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
023322  105367  000100                  DECB    $OMODE          ;;TYPE THIS DIGIT?
023326  100017                          BPL     7$              ;;BR IF NO
023330  042703  177770                  BIC     #177770,R3      ;;GET RID OF JUNK
023334  001002                          BNE     4$              ;;TEST FOR 0
023336  005704                          TST     R4              ;;SUPPRESS THIS 0?
023340  001403                          BEQ     5$              ;;BR IF YES
023342  005204                  4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
023344  052703  000060                  BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
023350  052703  000040          5$:     BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
023354  110367  000042                  MOVB    R3,8$           ;;SAVE FOR TYPING
023360  004567  176366                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
023364  023422                          .WORD   8$              ;ADDRESS OF MESSAGE TO BE TYPED
023366  105367  000032          7$:     DECB    $OCNT           ;;COUNT BY 1
023372  003346                          BGT     2$              ;;BR IF MORE TO DO
023374  002402                          BLT     6$              ;;BR IF DONE
023376  005204                          INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
023400  000743                          BR      2$              ;;GO DO THE LAST DIGIT
023402  012605                  6$:     MOV     (SP)+,R5        ;;RESTORE R5
023404  012604                          MOV     (SP)+,R4        ;;RESTORE R4
023406  012603                          MOV     (SP)+,R3        ;;RESTORE R3
023410  016666  000002  000004          MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
023416  012616                          MOV     (SP)+,(SP)
023420  000002                          RTI                     ;;RETURN
023422     000                  8$:     .BYTE   0               ;;STORAGE FOR ASCII DIGIT
023423     000                          .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
023424     000                  $OCNT:  .BYTE   0               ;;OCTAL DIGIT COUNTER
023425     000                  $OFILL: .BYTE   0               ;;ZERO FILL SWITCH
023426  000000                  $OMODE: .WORD   0               ;;NUMBER OF DIGITS TO TYPE
3618                                    ;ERROR TRAP SERVICE ROUTINE
3619 023430  005727                    ERRTRP: TST     (PC)+           ;CHECK IF PREV TRAP TO 4 REPORTED
3620 023432  000000                    1$:     .WORD   0               ;CONTAINS ERROR REPORTED FLAG
3621 023434  001010                            BNE     2$              ;BRANCH IF NOT REPORTED
3622 023436  005267  177770                    INC     1$              ;SET DOUBLE TRAP FLAG.
3623 023442  011667  155510                    MOV     (SP),   $IMP3   ;SAVE THE BAD PC FOR TYPOUT.
3624 023446  004767  174372                    JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
```

```
                023452  000031                          .WORD   31              ;ERROR TYPE CODE.
     3625       023454  000401                          BR      3$              ;SKIP HALT
     3626       023456  000000                  2$:     HALT                    ;ERROR! SECOND TRAP TO 4 OCCURRED
     3627                                                                        ;BEFORE FIRST WAS PRINTED
     3628       023460  005067  177746          3$:     CLR     1$
     3629       023464  000002                          RTI                     ;RETURN TO PROGRAM AND TRY TO RECOVER
     3630
     3631                                       .SBTTL  PHYSICAL ADDRESS TYPE ROUTINE
     3632                                       ;*  ROUTINE TO TYPE A PHYSICAL ADDRESS (22 BITS).
     3633       023466                  $TYPAD:
                023466  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
                023470  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
                023472  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
                023474  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
     3634       023476  016602  000012                  MOV     12(SP), R2      ;GET BASE ADDRESS
     3635       023502  005003                          CLR     R3              ;WORKING & INDEX REGISTER
     3636       023504  005767  155026                  TST     MMAVA           ;CHECK FOR MEM MGMT AVAILABLE
     3637       023510  001430                          BEQ     1$              ;BRANCH IF NO MEM MGMT
     3638       023512  032737  000001  177572          BIT     #1,     @#SR0   ;CHECK IF MEM MGMT ENABLED
     3639       023520  001424                          BEQ     1$              ;BRANCH IF MEM MGMT NOT ENABLED
     3640       023522  010201                          MOV     R2,     R1      ;COPY VIRTUAL ADR
     3641       023524  006101                          ROL     R1              ;SHUFFLE BITS 13,14,15 INTO 1,2,3
     3642       023526  006101                          ROL     R1
     3643       023530  006101                          ROL     R1
     3644       023532  006101                          ROL     R1
     3645       023534  006101                          ROL     R1
     3646       023536  042701  177761                  BIC     #177761, R1     ;CLR ALL EXCEPT BITS 1,2,3
     3647       023542  062701  172340                  ADD     #KIPAR0, R1     ;SET TO APPROPRIATE PAR
     3648       023546  011101                          MOV     (R1),   R1      ;GET CONTENTS OF PAR
     3649       023550  012700  000006                  MOV     #6,     R0      ;SET UP COUNTER
     3650       023554  006301                  4$:     ASL     R1              ;SHIFT PAR
     3651       023556  006103                          ROL     R3              ;SAVE OVERFLOW BITS
     3652       023560  077003                          SOB     R0,     4$      ;COUNT SIX SHIFTS
     3653       023562  042702  160000                  BIC     #160000, R2     ;SAVE BANK BITS
     3654       023566  060102                          ADD     R1,     R2      ;COMPUTE PHYSICAL ADDRESS
     3655       023570  005503                          ADC     R3              ;MAKE SURE CARRY ISN'T LOST!
     3656       023572  006302                  1$:     ASL     R2              ;FIRST DIGIT TO R3
     3657       023574  006103                          ROL     R3
     3658       023576  006002                          ROR     R2              ;RESTORE R2 FOR BITS 14-0
     3659       023600  010346                          MOV     R3,-(SP)        ;;SAVE R3 FOR TYPEOUT
                                                                                 ;;TYPE ADDRESS BITS 21-15
                                               ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOS ROUTINE
                                               ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
                023602  106746                      MFPS    -(SP)           ;PUT THE PROCESSOR STATUS ON THE STACK
                023604  105066  000001              CLRB    1(SP)           ;HIGH BYTE CLEARED TO INSURE KERNEL MODE
                                                                             ;ON PSW RETURN.
                023610  004767  177364              JSR     PC,     $TYPOS  ;GO TO THE SUBROUTINE
                023614     003                      .BYTE   3               ;;TYPE 3 DIGIT(S)
                023615     001                      .BYTE   1               ;;TYPE LEADING ZEROS
     3660       023616  010246                      MOV     R2,-(SP)        ;;SAVE R2 FOR TYPEOUT
                                                                             ;;TYPE ADDRESS BITS 14-0
                                               ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOS ROUTINE
                                               ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
                023620  106746                      MFPS    -(SP)           ;PUT THE PROCESSOR STATUS ON THE STACK
                023622  105066  000001              CLRB    1(SP)           ;HIGH BYTE CLEARED TO INSURE KERNEL MODE
                                                                             ;ON PSW RETURN.
                023626  004767  177346              JSR     PC,     $TYPOS  ;GO TO THE SUBROUTINE
```

```
        023632     005                          .BYTE   5               ;;TYPE 5 DIGIT(S)
        023633     001                          .BYTE   1               ;;TYPE LEADING ZEROS
  3661  023634  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
        023636  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
        023640  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
        023642  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
  3662  023644  012616                          MOV     (SP)+, (SP)     ;ADJUST THE STACK TO CLEAR DATA
  3663  023646  000207                          RTS     PC              ;RETURN
  3664
  3665                                          .SBTTL  STANDARD PROGRAM MESSAGES
  3666                                  ;;**************************************************************
  3667                                  ;VARIOUS MESSAGE PRINTOUTS USED THRUOUT
  3668                                  ;THE PROGRAM
  3669                                  ;;**************************************************************
  3670  023650     015     012     113  MMAMES: .ASCIZ  <15><12>'KT11 (MEMORY MANAGEMENT) AVAILABLE'
        023653     124     061     061
        023656     040     050     115
        023661     105     115     117
        023664     122     131     040
        023667     115     101     116
        023672     101     107     105
        023675     115     105     116
        023700     124     051     040
        023703     101     126     101
        023706     111     114     101
        023711     102     114     105
        023714     000
  3671  023715     015     012     062  AVAL22: .ASCIZ  <15><12>'22 BIT ADR AVAIL'<15><12>
        023720     062     040     102
        023723     111     124     040
        023726     101     104     122
        023731     040     101     126
        023734     101     111     114
        023737     015     012     000
  3672  023742     015     012     115  MEMMES: .ASCIZ  <15><12>'MEMORY MAP:'
        02    ,5   105     115     117
        023750     122     131     040
        023753     115     101     120
        023756     072     000
  3673  023760     015     012     120  MTMAP:  .ASCIZ  <15><12>'PARITY MEMORY MAP:'
        023763     101     122     111
        023766     124     131     040
        023771     115     105     115
        023774     117     122     131
        023777     040     115     101
        024002     120     072     000
  3674  024005     015     012     106  FROM:   .ASCIZ  <15><12>'FROM '
        024010     122     117     115
        024013     040     000
  3675  024015     040     124     117  TO:     .ASCIZ  ' TO '
        024020     040     000
  3676  024022     015     012     111  INSUFF: .ASCIZ  <15><12>'INSUFFICIENT MEMORY...FIRST 16K NOT ALL THERE!'
        024025     116     123     125
        024030     106     106     111
        024033     103     111     105
        024036     116     124     040
        024041     115     105     115
```

```
        024044  117     122     131
        024047  056     056     056
        024052  106     111     122
        024055  123     124     040
        024060  061     066     113
        024063  040     116     117
        024066  124     040     101
        024071  114     114     040
        024074  124     110     105
        024077  122     105     041
        024102  000
 3677   024103  015     012     116  MTR:     .ASCIZ  <15><12>'NO PARITY REGISTERS FOUND'
        024106  117     040     120
        024111  101     122     111
        024114  124     131     040
        024117  122     105     107
        024122  111     123     124
        024125  105     122     123
        024130  040     106     117
        024133  125     116     104
        024136  000
 3678   024137  015     012     122  PWRMSG:  .ASCIZ  <15><12>'RESTARTING AFTER A POWER FAILURE'<15><12>
        024142  105     123     124
        024145  101     122     124
        024150  111     116     107
        024153  040     101     106
        024156  124     105     122
        024161  040     101     040
        024164  120     117     127
        024167  105     122     040
        024172  106     101     111
        024175  114     125     122
        024200  105     015     012
        024203  000
 3679   024204  015     012     116  NOPES:   .ASCIZ  <15><12>'NO PARITY ERRORS FOUND ON MEMORY SCAN'<15><12>
        024207  117     040     120
        024212  101     122     111
        024215  124     131     040
        024220  105     122     122
        024223  117     122     123
        024226  040     106     117
        024231  125     116     104
        024234  040     117     116
        024237  040     115     105
        024242  115     117     122
        024245  131     040     123
        024250  103     101     116
        024253  015     012     000
 3680   024256  015     012     120  PROREL:  .ASCII  <15><12>'PROGRAM NOW RESIDES BACK AT 0 TO 8K'
        024261  122     117     107
        024264  122     101     115
        024267  040     116     117
        024272  127     040     122
        024275  105     123     111
        024300  104     105     122
        024303  040     102     101
        024306  103     113     040
```

```
                024311  101     124     040
                024314  060     040     124
                024317  117     040     070
                024322  113
3681            024323  015     012     110             .ASCIZ  <15><12>'HIT CONTINUE FOR NORMAL RUNNING'<15><12>
                024326  111     124     040
                024331  103     117     116
                024334  124     111     116
                024337  125     105     040
                024342  106     117     122
                024345  040     116     117
                024350  122     115     101
                024353  114     040     122
                024356  125     116     116
                024361  111     116     107
                024364  015     012     000
3682            024367  015     012     120     MX1:    .ASCIZ  <15><12>'PARITY REGISTER AT '
                024372  101     122     111
                024375  124     131     040
                024400  122     105     107
                024403  111     123     124
                024406  105     122     040
                024411  101     124     040
                024414  000
3683            024415  040     103     117     MX2:    .ASCIZ  ' CONTROLS '
                024420  116     124     122
                024423  117     114     123
                024426  040     000
3684            024430  015     012     116     NOMEM:  .ASCIZ  <15><12>'NO MEMORY FOUND.'
                024433  117     040     115
                024436  105     115     117
                024441  122     131     040
                024444  106     117     125
                024447  116     104     056
                024452  000
3685            024453  015     012     012     FADMES: .ASCII  <15><12><12><12>'INPUT ALL PARAMETERS IN OCTAL.'
                024456  012     111     116
                024461  120     125     124
                024464  040     101     114
                024467  114     040     120
                024472  101     122     101
                024475  115     105     124
                024500  105     122     123
                024503  040     111     116
                024506  040     117     103
                024511  124     101     114
                024514  056
3686            024515  015     012     106             .ASCIZ  <15><12>'FIRST ADDRESS:  '
                024520  111     122     123
                024523  124     040     101
                024526  104     104     122
                024531  105     123     123
                024534  072     040     040
                024537  000
3687            024540  015     012     114     LADMES: .ASCIZ  <15><12>'LAST ADDRESS:   '
                024543  101     123     124
                024546  040     101     104
```

```
                024551  104     122     105
                024554  123     123     072
                024557  040     040     040
                024562  000
 3688           024563  015     012     077     BADADR: .ASCIZ  <15><12>'?ADDRESS IN UNMAPPED BANK?'
                024566  101     104     104
                024571  122     105     123
                024574  123     040     111
                024577  116     040     125
                024602  116     115     101
                024605  120     120     105
                024610  104     040     102
                024613  101     116     113
                024616  077     000
 3689           024620  015     012     123     CONST:  .ASCIZ  <15><12>'SELECT CONSTANT:'
                024623  105     114     105
                024626  103     124     040
                024631  103     117     116
                024634  123     124     101
                024637  116     124     072
                024642  000
 3690           024643  015     012     125     UNEXPT: .ASCIZ  <15><12>'UNEXPECTED MEMORY PARITY ERROR'
                024646  116     105     130
                024651  120     105     103
                024654  124     105     104
                024657  040     115     105
                024662  115     117     122
                024665  131     040     120
                024670  101     122     111
                024673  124     131     040
                024676  105     122     122
                024701  117     122     000
 3691           024704  015     012     120     PRELOC: .ASCIZ  <15><12>'PROGRAM RELOCATED TO '
                024707  122     117     107
                024712  122     101     115
                024715  040     122     105
                024720  114     117     103
                024723  101     124     105
                024726  104     040     124
                024731  117     040     000
 3692           024734  015     012     115     MTOE:   .ASCIZ  <15><12>'MORE THAN ONE PARITY ERROR FOUND.'
                024737  117     122     105
                024742  040     124     110
                024745  101     116     040
                024750  117     116     105
                024753  040     120     101
                024756  122     111     124
                024761  131     040     105
                024764  122     122     117
                024767  122     040     106
                024772  117     125     116
                024775  104     056     000
 3693           025000  015     012     123     SCANM:  .ASCIZ  <15><12>'SCANNING MEMORY FOR BAD PARITY.'
                025003  103     101     116
                025006  116     111     116
                025011  107     040     115
                025014  105     115     117
```

```
        025017  122     131     040
        025022  106     117     122
        025025  040     102     101
        025030  104     040     120
        025033  101     122     111
        025036  124     131     056
        025041  000
3694    025042  015     012     120  PEWNC:   .ASCIZ  <15><12>'PARITY ERROR WILL NOT CLEAR.'
        025045  101     122     111
        025050  124     131     040
        025053  105     122     122
        025056  117     122     040
        025061  127     111     114
        025064  114     040     116
        025067  117     124     040
        025072  103     114     105
        025075  101     122     056
        025100  000
3695    025101  015     012     116  NOMTST: .ASCIZ  <15><12>'NO MEMORY TESTED.'
        025104  117     040     115
        025107  105     115     117
        025112  122     131     040
        025115  124     105     123
        025120  124     105     104
        025123  056     000
3696    025125  015     012     123  SKPMES: .ASCIZ  <15><12>'SKIPPING TEST #'
        025130  113     111     120
        025133  120     111     116
        025136  107     040     124
        025141  105     123     124
        025144  040     043     000
3697    025147  377     377     000  FILL2:  .ASCIZ  <377><377>
3698
3699                                 .SBTTL  ERROR REPORTING MESSAGES AND TABLES.
3700                                 ;;*****************************************************************
3701                                 ;* MESSAGE BLOCK FOR ERROR TABLE TYPEOUTS
3702                                 ;;*****************************************************************
3703    025152  120     101     122  DM1:    .ASCIZ  'PARITY REGISTER DATA ERROR.'
        025155  111     124     131
        025160  040     122     105
        025163  107     111     123
        025166  124     105     122
        025171  040     104     101
        025174  124     101     040
        025177  105     122     122
        025202  117     122     056
        025205  000
3704    025206  101     104     104  DM2:    .ASCIZ  'ADDRESS TEST ERROR(TST1-5).'
        025211  122     105     123
        025214  123     040     124
        025217  105     123     124
        025222  040     105     122
        025225  122     117     122
        025230  050     124     123
        025233  124     061     055
        025236  065     051     056
        025241  000
```

```
3705 025242   103   117   116  DM4:   .ASCIZ  'CONSTANT DATA ERROR(TST6-10).'
     025245   123   124   101
     025250   116   124   040
     025253   104   101   124
     025256   101   040   105
     025261   122   122   117
     025264   122   050   124
     025267   123   124   066
     025272   055   061   060
     025275   051   056   000
3706 025300   122   117   124  DM5:   .ASCIZ  'ROTATING BIT ERROR(TST11-12).'
     025303   101   124   111
     025306   116   107   040
     025311   102   111   124
     025314   040   105   122
     025317   122   117   122
     025322   050   124   123
     025325   124   061   061
     025330   055   061   062
     025333   051   056   000
3707 025336   115   117   123  DM6:   .ASCIZ  'MOS REFRESH TEST ERROR (TST 24-25).'
     025341   040   122   105
     025344   106   122   105
     025347   123   110   040
     025352   124   105   123
     025355   124   040   105
     025360   122   122   117
     025363   122   040   050
     025366   124   123   124
     025371   040   062   064
     025374   055   062   065
     025377   051   056   000
3708 025402   106   101   124  DM7:   .ASCIZ  'FATAL ERROR HALT'
     025405   101   114   040
     025410   105   122   122
     025413   117   122   040
     025416   110   101   114
     025421   124   000
3709 025423   115   101   122  DM10:  .ASCIZ  "MARCHING 1'S AND 0'S ERROR(TST 23)."
     025426   103   110   111
     025431   116   107   040
     025434   061   047   123
     025437   040   101   116
     025442   104   040   060
     025445   047   123   040
     025450   105   122   122
     025453   117   122   050
     025456   124   123   124
     025461   040   062   063
     025464   051   056   000
3710 025467   120   101   122  DM11:  .ASCIZ  'PARITY MEMORY ADDRESS ERROR(TST13).'
     025472   111   124   131
     025475   040   115   105
     025500   115   117   122
     025503   131   040   101
     025506   104   104   122
     025511   105   123   123
```

```
                025514  040  105  122
                025517  122  117  122
                025522  050  124  123
                025525  124  061  063
                025530  051  056  000
3711            025533  104  101  124  DM12:   .ASCIZ  "DATIO WITH WRONG PARITY DIDN'T TRAP(TST13)."
                025536  111  117  040
                025541  127  111  124
                025544  110  040  127
                025547  122  117  116
                025552  107  040  120
                025555  101  122  111
                025560  124  131  040
                025563  104  111  104
                025566  116  047  124
                025571  040  124  122
                025574  101  120  050
                025577  124  123  124
                025602  061  063  051
                025605  056  000
3712            025607  127  122  117  DM13:   .ASCIZ  'WRONG PARITY DETECTED, BUT NO REGISTER SHOWS ERROR FLAG.'
                025612  116  107  040
                025615  120  101  122
                025620  111  124  131
                025623  040  104  105
                025626  124  105  103
                025631  124  105  104
                025634  054  040  102
                025637  125  124  040
                025642  116  117  040
                025645  122  105  107
                025650  111  123  124
                025653  105  122  040
                025656  123  110  117
                025661  127  123  040
                025664  105  122  122
                025667  117  122  040
                025672  106  114  101
                025675  107  056  000
3713            025700  120  101  122  DM14:   .ASCIZ  'PARITY REGISTER NOT MAPPED AS CONTROLLING THIS ADDRESS(TST13).'
                025703  111  124  131
                025706  040  122  105
                025711  107  111  123
                025714  124  105  122
                025717  040  116  117
                025722  124  040  115
                025725  101  120  120
                025730  105  104  040
                025733  101  123  040
                025736  103  117  116
                025741  124  122  117
                025744  114  114  111
                025747  116  107  040
                025752  124  110  111
                025755  123  040  101
                025760  104  104  122
                025763  105  123  123
```

```
        025766  050  124  123
        025771  124  061  063
        025774  051  056  000
3714    025777  115  117  122  DM16:   .ASCIZ  'MORE THAN ONE REGISTER INDICATED PARITY ERROR.'
        026002  105  040  124
        026005  110  101  116
        026010  040  117  116
        026013  105  040  122
        026016  105  107  111
        026021  123  124  105
        026024  122  040  111
        026027  116  104  111
        026032  103  101  124
        026035  105  104  040
        026040  120  101  122
        026043  111  124  131
        026046  040  105  122
        026051  122  117  122
        026054  056  000
3715    026056  104  101  124  DM17:   .ASCIZ  "DATA SHOULDN'T HAVE CHANGED WHEN PARITY ERROR TRAPPED(TST13)."
        026061  101  040  123
        026064  110  117  125
        026067  114  104  116
        026072  047  124  040
        026075  110  101  126
        026100  105  040  103
        026103  110  101  116
        026106  107  105  104
        026111  040  127  110
        026114  105  116  040
        026117  120  101  122
        026122  111  124  131
        026125  040  105  122
        026130  122  117  122
        026133  040  124  122
        026136  101  120  120
        026141  105  104  050
        026144  124  123  124
        026147  061  063  051
        026152  056  000
3716    026154  122  101  116  DM20:   .ASCIZ  'RANDOM DATA ERROR(TST14).'
        026157  104  117  115
        026162  040  104  101
        026165  124  101  040
        026170  105  122  122
        026173  117  122  050
        026176  124  123  124
        026201  061  064  051
        026204  056  000
3717    026206  111  116  123  DM21:   .ASCIZ  'INSTRUCTION EXECUTION ERROR(TST15-22).'
        026211  124  122  125
        026214  103  124  111
        026217  117  116  040
        026222  105  130  105
        026225  103  125  124
        026230  111  117  116
        026233  040  105  122
```

```
                026236  122     117     122
                026241  050     124     123
                026244  124     061     065
                026247  055     062     062
                026252  051     056     000
3718            026255  120     122     117     DM23:   .ASCIZ  'PROGRAM CODE CHANGED WHEN RELOCATED.'
                026260  107     122     101
                026263  115     040     103
                026266  117     104     105
                026271  040     103     110
                026274  101     116     107
                026277  105     104     040
                026302  127     110     105
                026305  116     040     122
                026310  105     114     117
                026313  103     101     124
                026316  105     104     056
                026321  000
3719            026322  124     122     101     DM24:   .ASCIZ  'TRAPPED, BUT NO REGISTER HAD ERROR BIT SET.'
                026325  120     120     105
                026330  104     054     040
                026333  102     125     124
                026336  040     116     117
                026341  040     122     105
                026344  107     111     123
                026347  124     105     122
                026352  040     110     101
                026355  104     040     105
                026360  122     122     117
                026363  122     040     102
                026366  111     124     040
                026371  123     105     124
                026374  056     000
3720            026376  124     122     101     DM25:   .ASCIZ  'TRAPPED TO 114.'
                026401  120     120     105
                026404  104     040     124
                026407  117     040     061
                026412  061     064     056
                026415  000
3721            026416  106     101     111     DM26:   .ASCIZ  'FAILED TO TRAP.'
                026421  114     105     104
                026424  040     124     117
                026427  040     124     122
                026432  101     120     056
                026435  000
3722            026436  050     101     103     DM27:   .ASCIZ  "(ACTION ENABLE WASN'T SET)."
                026441  124     111     117
                026444  116     040     105
                026447  116     101     102
                026452  114     105     040
                026455  127     101     123
                026460  116     047     124
                026463  040     123     105
                026466  124     051     056
                026471  000
3723            026472  015     012     124     DM31:   .ASCIZ  <15><12>'TRAPPED TO 4
                026475  122     101     120
```

```
        026500  120  105  104
        026503  040  124  117
        026506  040  064  040
        026511  000

3724
3725                                    ;;*****************************************************************
3726                                    ;DATA COLUMN HEADINGS
3727                                    ;; *****************************************************************
3728
3729    026512  120  103  011  DH1:     .ASCIZ  'PC     REG       S/B     WAS'
        026515  122  105  107
        026520  011  040  040
        026523  123  057  102
        026526  011  040  040
        026531  127  101  123
        026534  000
3730    026535  126  057  120  DH2:     .ASCIZ  'V/PC   P/PC      MA        S/B     WAS'
        026540  103  011  120
        026543  057  120  103
        026546  011  040  040
        026551  115  101  011
        026554  040  040  040
        026557  040  123  057
        026562  102  011  040
        026565  040  040  040
        026570  127  101  123
        026573  000
3731    026574  126  057  120  DH12:    .ASCIZ  'V/PC   P/PC      MA        S/B'
        026577  103  011  120
        026602  057  120  103
        026605  011  040  040
        026610  115  101  011
        026613  040  040  040
        026616  040  123  057
        026621  102  000
3732    026623  126  057  120  DH14:    .ASCIZ  'V/PC   P/PC      REG       MA'
        026626  103  011  120
        026631  057  120  103
        026634  011  040  040
        026637  122  105  107
        026642  011  040  040
        026645  040  040  115
        026650  101  000
3733    026652  126  057  120  DH15:    .ASCIZ  'V/PC   P/PC      MAUT      REG       S/B     WAS'
        026655  103  011  120
        026660  057  120  103
        026663  011  040  040
        026666  115  101  125
        026671  124  011  040
        026674  040  040  040
        026677  122  105  107
        026702  040  040  040
        026705  040  040  040
        026710  040  123  057
        026713  102  040  040
        026716  040  040  040
        026721  127  101  123
```

```
                026724  000
3734            026725  126     057     120  DH21:   .ASCIZ  'V/PC   P/PC    IUT     MA      S/B     WAS'
                026730  103     011     120
                026733  057     120     103
                026736  011     040     040
                026741  111     125     124
                026744  011     040     040
                026747  115     101     011
                026752  040     040     040
                026755  040     123     057
                026760  102     011     040
                026763  040     040     040
                026766  127     101     123
                026771  000
3735            026772  126     057     120  DH23:   .ASCIZ  'V/PC   P/PC    SRC MA  DST MA  S/B     WAS'
                026775  103     011     120
                027000  057     120     103
                027003  011     040     040
                027006  123     122     103
                027011  040     115     101
                027014  040     040     040
                027017  040     104     123
                027022  124     040     115
                027025  101     040     040
                027030  040     040     123
                027033  057     102     040
                027036  040     040     040
                027041  040     127     101
                027044  123     000
3736            027046  126     057     120  DH24:   .ASCIZ  'V/PC   P/PC    TRP/PC'
                027051  103     011     120
                027054  057     120     103
                027057  011     040     040
                027062  124     122     120
                027065  057     120     103
                027070  000
3737            027071  126     057     120  DH25:   .ASCIZ  'V/PC   P/PC    TRP/PC  REG     WAS'
                027074  103     011     120
                027077  057     120     103
                027102  011     040     040
                027105  124     122     120
                027110  057     120     103
                027113  040     040     040
                027116  040     122     105
                027121  107     040     040
                027124  040     040     040
                027127  040     040     127
                027132  101     123     000
3738            027135  126     057     120  DH26:   .ASCIZ  'V/PC   P/PC    REG     WAS'
                027140  103     011     120
                027143  057     120     103
                027146  011     040     040
                027151  122     105     107
                027154  011     040     040
                027157  040     040     127
                027162  101     123     000
3739            027165  122     105     107  DH30:   .ASCIZ  'REG     WAS     MA      WAS'
```

```
        027170  011     040     040
        027173  127     101     123
        027176  011     040     040
        027201  115     101     011
        027204  040     040     040
        027207  040     127     101
        027212  123     000
3740    027214  126     057     120     DH31:   .ASCIZ  'V/PC   P/PC      MA        S/B  WAS'
        027217  103     011     120
        027222  057     120     103
        027225  011     040     040
        027230  115     101     011
        027233  040     040     040
        027236  040     123     057
        027241  102     040     040
        027244  127     101     123
        027247  000
3741
3742                                    ;;*****************************************************************
3743                                    ;* DATA FORMAT TABLE FOR ERROR PRINTOUT.
3744                                    ;;*****************************************************************
3745    027250  000     377     000     DF1:    .BYTE   0,-1,0,0
        027253  000
3746    027254  000     377     377     DF2:    .BYTE   0,-1,-1,0,0
        027257  000     000
3747    027261  000     377     377     DF3:    .BYTE   0,-1,-1,-2,-2
        027264  376     376
3748    027266  000     377     377     DF14:   .BYTE   0,-1,-1,-1,0,0
        027271  377     000     000
3749    027274  000     377     000     DF21:   .BYTE   0,-1,0,-1,0,0
        027277  377     000     000
3750    027302  377     000     377     DF30:   .BYTE   -1,0,-1,-2
        027305  376
3751                                            .EVEN
3752
3753            032000                  . = 32000                       ;THE LOADERS ARE SAVE HERE TO END OF 8K
3754                                                                    ;1536. WORDS
3755
3756            000001                          .END
```

```
ABASE = 000000   BIT0  = 000001   DIDO   011350   GMPRBA 005334   MPR0   002272
ACDW1 = 000000   BIT00 = 000001   DIPDO  011606   GMPRC  005340   MPR1   002336
ACDW2 = 000000   BIT01 = 000002   DISMAP 005160   GMPRD  005376   MPR10  003042
ACPUOP= 000000   BIT02 = 000004   DISPLA 001132   HT     = 000011   MPR11  003106
ADDW0 = 000000   BIT03 = 000010   DISPRE 000174   INITDN 013254   MPR12  003152
ADDW1 = 000000   BIT04 = 000020   DM1    025152   INITEX 013362   MPR13  003216
ADDW10= 000000   BIT05 = 000040   DM10   025423   INITMM 013156   MPR14  003262
ADDW11= 000000   BIT06 = 000100   DM11   025467   INSUFF 024022   MPR15  003326
ADDW12= 000000   BIT07 = 000200   DM12   025533   IOTVEC= 000020   MPR2   002402
ADDW13= 000000   BIT08 = 000400   DM13   025607   KDJDA  002000   MPR3   002446
ADDW14= 000000   BIT09 = 001000   DM14   025700   KIPAR0= 172340   MPR4   002512
ADDW15= 000000   BIT1  = 000002   DM16   025777   KIPAR1= 172342   MPR5   002556
ADDW2 = 000000   BIT10 = 002000   DM17   026056   KIPAR2= 172344   MPR6   002622
ADDW3 = 000000   BIT11 = 004000   DM2    025206   KIPAR3= 172346   MPR7   002666
ADDW4 = 000000   BIT12 = 010000   DM20   026154   KIPAR4= 172350   MPR8   002732
ADDW5 = 000000   BIT13 = 020000   DM21   026206   KIPAR5= 172352   MPR9   002776
ADDW6 = 000000   BIT14 = 040000   DM23   026255   KIPAR6= 172354   MTMAP  023760
ADDW7 = 000000   BIT15 = 100000   DM24   026322   KIPAR7= 172356   MTOE   024734
ADDW8 = 000000   BIT2  = 000004   DM25   026376   KIPDR0= 172300   MTR    024103
ADDW9 = 000000   BIT3  = 000010   DM26   026416   KIPDR1= 172302   MX1    024367
ADEVCT= 000000   BIT4  = 000020   DM27   026436   KIPDR2= 172304   MX2    024415
ADEVM = 000000   BIT5  = 000040   DM31   026472   KIPDR3= 172306   NOMEM  024430
AE    = 000001   BIT6  = 000100   DM4    025242   KIPDR4= 172310   NOMTST 025101
AENV  = 000000   BIT7  = 000200   DM5    025300   KIPDR5= 172312   NONKT  004646
AENVM = 000000   BIT8  = 000400   DM6    025336   KIPDR6= 172314   NOPES  024204
AFATAL= 000000   BIT9  = 001000   DM7    025402   KIPDR7= 172316   OEFLG  001760
AMADR1= 000000   BNKTAB 017526   DONE   012502   KTSIZ  004760   PARMAT 016146
AMADR2= 000000   BPTVEC= 000014   DPDBH  011756   LADMES 024540   PARVEC= 000114
AMADR3= 000000   CHGG1  003434   DPDBL  011672   LDDISP 001512   PBTRP  010520
AMADR4= 000000   CHGG2  016040   DSWR   = 177570   LF     = 000012   PBTRP1 010530
AMAMS1= 000000   CKPMER 016040   DT1    002042   LMAD   001510   PESRV  015632
AMAMS2= 000000   CLRPAR 016672   DT12   002070   LMEMHI 017524   PEWNC  025042
AMAMS3= 000000   CONST  024620   DT14   002102   LMEMLO 017522   PHYADR 014276
AMAMS4= 000000   CR     = 000015   DT15   002114   LSIFLG 001776   PIRQ   - 177772
AMSGAD= 000000   CRLF   = 000200   DT2    002054   MAMF   015740   PIRQVE= 000240
AMSGLG= 000000   CTRLS  006060   DT21   002132   MAMF1  016024   PLUS1  002002
AMSGTY= 000000   DDISP  = 177570   DT23   002150   MAMF2  016036   PMEMAP 001654
AMTYP1= 000000   DF1    027250   DT24   002166   MANUAL 006350   PRELOC 024704
AMTYP2= 000000   DF14   027266   DT25   002176   MANUL1 006366   PRGMAP 000534
AMTYP3= 000000   DF2    027254   DT26   002212   MANUL2 006770   PROREL 024256
AMTYP4= 000000   DF21   027274   DT30   002224   MAPRB  005432   PR0    - 000000
APASS = 000000   DF3    027261   DT31   002236   MASK4K= 017777   PR1    = 000040
APRIOR= 000000   DF30   027302   EMTVEC= 000030   MASK8K= 037777   PR2    - 000100
APTCSU= 000040   DH1    026512   ENDBKT 017530   MEMMAP 001514   PR3    = 000140
APTENV= 000001   DH12   026574   ENDINS 020026   MEMMES 023742   PR4    = 000200
APTSIZ= 000200   DH14   026623   ERROR  = 104000   MFPT   = 000007   PR5    - 000240
APTSPO= 000100   DH15   026652   ERRTRP 023430   MMAMES 023650   PR6    - 000300
ASWREG= 000000   DH2    026535   ERRVEC= 000004   MMAVA  000536   PR7    - 000340
ATESTN= 000000   DH21   026725   FADMES 024453   MMDOWN 013734   PS     = 177776
AUNIT = 000000   DH23   026772   FILL2  025147   MMINIT 013000   PSCAN  016212
AUSWR = 000000   DH24   027046   FLAG8K 001757   MMORE  001754   PSW    = 177776
AVAL22  023715   DH25   027071   FLG30K 001774   MMUP   013370   PWRMSG 024137
AVECT1= 000000   DH26   027135   FROM   024005   MMVEC  = 000250   PWRVEC- 000024
AVECT2= 000000   DH30   027165   FSTADR 001762   MPAMEM 005404   RADTAB 002004
BADADR  024563   DH31   027214   GMPR   005262   MPEND  002270   RANTST 011232
BANKNO  014374   DIDBH  011516   GMPRA  005310   MPPATS 002244   RELOC  014626
BITP    001714   DIDBL  011432   GMPRB  005324   MPRX   003372   RELOCF 000532
```

```
RFLTOP  014750        SW10    002000        WWPB3   011072        $ENVM   001215        $OVER   020014
RELO    015264        SW11  - 004000        WWPB4   011162        $EOP    012624        $PASS   001202
RESCHK  006232        SW12  = 010000        WWPB5   011206        $EOPCT  012646        $PASTM  001326
RESLDR  015464        SW13  = 020000        WWPB6   011216        $ERFLG  001073        $PRINT  021752
RESRVD  001506        SW14    040000        W3X9    014546        $ERMAX  001105        $PWRAD  000702
RESTAR  000300        SW15  - 100000        $APTHD  001320        $ERROR  020044        $PWRDN  000540
RESTOR  000304        SW2   = 000004        $APTR   001502        $ERRPC  001106        $PWRMG  000676
RESTI   000306        SW3   = 000010        $ASTAT  001334        $ERRTB  003434        $PWRUP  000612
RESTI2  000324        SW4   = 000020        $ASTEN  001500        $ERRTY  020274        $QUES   001170
RESVEC  000010        SW5   = 000040        $ATYC   022364        $ERTTL  001102        $RDCHR  021114
ROTATE  014500        SW6   = 000100        $ATY1   022340        $ESCAP  001162        $RDLIN  021244
RW      000006        SW7   = 000200        $ATY3   022346        $ETABL  001214        $RDOCT  021576
R6    %000006         SW8   = 000400        $ATY4   022356        $ETEND  001320        $RDSZ - 000010
R7    %000007         SW9   = 001000        $ATY6   022654        $FATAL  001176        $SAVR6  000712
SAVLDR  015552        TBITVE= 000014        $ATY7   022720        $FFLG   022744        $SCOPE  017532
SAVTST  001614        TEMP    001770        $AUTOB  001124        $FILLC  001146        $SETUP- 000130
SCAN    025000        TEMP1   001772        $BASE   001250        $FILLS  001145        $STUP = 177777
SCOPE - 000004        TKVEC = 000060        $BDADR  001112        $GDADR  001110        $SVLAD  017760
SECT2   007612        TMAP    005654        $BDDAT  001116        $GDDAT  001114        $SVPC = 000010
SELECT  003752        TO      024015        $BELL   001164        $GET42  012712        $SWR  = 167400
SELFLG  001756        TPVEC = 000064        $CDW1   001254        $GTSWR  020660        $SWREG  001216
SETAE   016004        TRAPVE= 000034        $CDW2   001256        $HD   = 000000        $SWRMK= 000340
SETCON  014462        TRTVEC= 000014        $CHARC  022334        $HIBTS  001320        $TESTN  001200
SKPMES  025125        TSTMAP  001554        $CKSWR  020606        $HIOC   021750        $TIMES  001160
SPREND  017024        TST1    007102        $CMTAG  001070        $ICNT   001074        $TKB    001136
SPRNT   016714        TST10   010004        $CM3  = 000000        $ILLUP  000706        $TKS    001134
SPRNTA  017014        TST11   010062        $CM4  = 000004        $INTAG  001125        $TMP0   001150
SPRNTB  017020        TST12   010142        $CNTLG  021547        $ITEMB  001104        $TMP1   001152
SPRNTP  016752        TST13   010220        $CNTLU  021542        $LF     001172        $TMP2   001154
SPRNTQ  016726        TST14   011226        $CPUOP  001222        $LFLG   022743        $TMP3   001156
SPRNT0  016756        TST15   011344        $CRLF   001171        $LPADR  001076        $TN   = 000026
SPRNT1  016764        TST16   011426        $DBLK   023170        $LPERR  001100        $TPB    001142
SPRNT2  017002        TST17   011512        $DDW0   001260        $MADR1  001226        $TPFLG  001147
SPRNT3  016776        TST2    007224        $DDW1   001262        $MADR2  001232        $TPS    001140
SR0   = 177572        TST20   011602        $DDW10  001304        $MADR3  001236        $TSTM   001324
SR1   = 177574        TST21   011666        $DDW11  001306        $MADR4  001242        $TSTNM  001072
SR2   = 177576        TST22   011752        $DDW12  001310        $MAIL   001174        $TTYIN  021532
SR3   = 172516        TST23   012040        $DDW13  001312        $MAMS1  001224        $TYPAD  023466
STACK - 001100        TST24   012272        $DDW14  001314        $MAMS2  001230        $TYPDS  022746
START   003744        TST25   012376        $DDW15  001316        $MAMS3  001234        $TYPE   022002
STARTA  003760        TST3    007324        $DDW2   001264        $MAMS4  001240        $TYPEC  022216
START1  007006        TST32   012506        $DDW3   001266        $MBADR  001322        $TYPEX  022336
STKLMT= 177774        TST4    007432        $DDW4   001270        $MFLG   022742        $TYPOC  023224
SWR     001130        TST5    007520        $DDW5   001272        $MNEW   021565        $TYPON  023240
SWREG   000176        TST6    007652        $DDW6   001274        $MSGAD  001210        $TYPOS  023200
SW0   = 000001        TST6A   007656        $DDW7   001276        $MSGLG  001212        $UNIT   001206
SW00  = 000001        TST7    007676        $DDW8   001300        $MSGTY  001174        $UNITM  001330
SW01  = 000002        TYPMAP  017026        $DDW9   001302        $MSWR   021554        $USWR   001220
SW02  = 000004        UNEXPT  024643        $DEVCT  001204        $MTYP1  001225        $VECT1  001244
SW03  = 000010        UP    = 000000        $DEVM   001252        $MTYP2  001231        $VECT2  001246
SW04  = 000020        VMK     013060        $DOAGN  012754        $MTYP3  001235        $VERPC  001504
SW05  = 000040        VMK1    013132        $DTBL   023160        $MTYP4  001241        $XOFF = 000023
SW06  = 000100        WWP     001766        $ENDAD  012722        $MXCNT  020030        $XON  = 000021
SW07  = 000200        WWPBYT  010262        $ENDCT  012654        $NULL   001144        $XTSTR  017556
SW08  = 000400        WWPB0   010224        $ENDMG  012760        $NWTST- 000001        $OFILL  023425
SW09  = 001000        WWPB1   010322        $ENULL  012775        $OCNT   023424        .BITPT  002016
SW1   = 000002        WWPB2   010366        $ENV    001214        $OMODE  023426        .CONST  001764
```

Symbol table SEQ 0120

```
.EIGHT  002036          .MPPAT  002030          .PBTRP  002026          .RESRV  002020          .TSTMA  002012
.ERRTB  002034          .MPRX   002024          .PESRV  002032          .SAVTS  002010          .TST32  002040
.MEMMA  002006          .MPRO   002022          .PMEMA  002014          .STACK  002004          .$X     = 001320

. ABS.  032000    000   (RW,I,GBL,ABS,OVR)
        000000    001   (RW,I,LCL,REL,CON)
Errors detected:  0

*** Assembler statistics

Work file reads: 655
Work file writes: 514
Size of work file: 65536 Words  ( 0 Pages)
Size of core pool: 19714 Words  ( 75 Pages)
Operating system: RSX-11M/PLUS (Under VAX/VMS)

Elapsed time: 00:03:03.04
CVMSAB.BIN,CVMSAB.LST=SYSMAC.MLB/ML,CVMSAB.P11
```