

MSV11-DD,LF,
PK,ED,LK,PF,PL

0-2 MEG MEM EXER
CVMSAAO

AH-S437A-MC
FICHE 1 OF 1

OCT 1981
COPYRIGHT © 1981
MADE IN USA



.REM ^

IDENTIFICATION
=====

PRODUCT CODE: AC-S435A-MC
PRODUCT NAME: CVMSAAO 0-2 MEG MEM EXER
PRODUCT DATE: APRIL, 1981
MAINTAINER: DIAGNOSTIC ENGINEERING

THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION. DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR ANY ERRORS THAT MAY APPEAR IN THIS DOCUMENT.

NO RESPONSIBILITY IS ASSUMED FOR THE USE OR RELIABILITY OF SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL OR ITS AFFILIATED COMPANIES.

COPYRIGHT (C) 1981 BY DIGITAL EQUIPMENT CORPORATION

THE FOLLOWING ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION:

DIGITAL PDP UNIBUS MASSBUS
DEC DECUS DECTAPE

REVISION HISTORY
=====

REVISION A:

APRIL 1981

TABLE OF CONTENTS

1.0	GENERAL PROGRAM INFORMATION.
1.1	PROGRAM PURPOSE (ABSTRACT)
1.2	SYSTEM REQUIREMENTS
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.

1.1 PROGRAM PURPOSE (ABSTRACT)

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

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

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

1.2 SYSTEM REQUIREMENTS

A. HARDWARE REQUIREMENTS

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

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

B. SOFTWARE REQUIREMENTS

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

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

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

1.2.1 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.

1.3 RELATED DOCUMENTS AND STANDARDS

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

1.4 DIAGNOSTIC HIERARCHY PREREQUISITES

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

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

1.5 ASSUMPTIONS

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

2.0 OPERATING INSTRUCTIONS

2.1 LOADING AND STARTING PROCEDURES

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

2.1.2 STARTING ADDRESS 200:
NORMAL PROGRAM EXECUTION.

2.1.3 STARTING ADDRESS 204:
RESTART PROGRAM USING PREVIOUSLY SELECTED PARAMETERS.

2.2 SPECIAL ENVIRONMENTS

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

2.3 PROGRAM OPTIONS

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

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

2.4 EXECUTION TIMES

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

A. FOR NON-PARITY MEMORY

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

B. FOR PARITY MEMORY

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

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

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

3.0 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

```

-----
| I | I | I | I | I | I | I | I | I | I | I | I | I | I | I | I |
| PE |   |   |   |   |   |   |   |   |   |   |   | WP | AE |
| I | I | I | I | I | I | I | I | I | I | I | I | I | I | I | I |
-----
15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00

```

BIT ASSIGNMENTS ARE DEFINED AS FOLLOWS:

BIT15 PARITY ERROR

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

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

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

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

6.0 SUB-TEST SUMMARIES

6.1 SECTION 1: ADDRESS TESTS.

THESE TESTS VERIFY THE UNIQUENESS OF EVERY MEMORY ADDRESS.

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

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

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

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

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

6.2 SECTION 2: WORST CASE NOISE TESTS.

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

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

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

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

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

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

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

6.3 SECTION 3: INSTRUCTION EXECUTION TESTS.

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

THAT THEY EXECUTED CORRECTLY.

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

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

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

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

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

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

6.4 SECTION 4: MOS TESTS

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

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

6.5 SPECIAL TOGGLE IN TESTS

6.5.1 TOGGLE-IN-PROGRAM #1

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

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

```

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

```

6.5.2 TOGGLE-IN-PROGRAM #2

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

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

```

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

```

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

```

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

```

7.0 PROGRAM LISTING ATTACHED

14	OPERATIONAL SWITCH SETTINGS
29	BASIC DEFINITIONS
141	MEMORY MANAGEMENT DEFINITIONS
189	TRAP CATCHER
198	STARTING ADDRESS(ES)
206	ACT11 HOOKS
282	POWER DOWN AND UP ROUTINES
324	COMMON TAGS
372	APT MAILBOX-ETABLE
438	APT PARAMETER BLOCK
460	APT STATISTICS TABLE
670	MEMORY PARITY PATTERNS TABLE
688	MEMORY PARITY REGISTER ADDRESS TABLE
993	ERROR POINTER TABLE
1138	START: SETUP AND MAP MEMORY
1149	INITIALIZE THE COMMON TAGS
1183	TYPE PROGRAM NAME
1191	GET VALUE FOR SOFTWARE SWITCH REGISTER
1410	MAP PARITY REGISTERS
1446	MAP PARITY MEMORY
1529	DISPLAY PARITY MEMORY MAP
1608	TEST PARITY REGISTERS
1824	SECTION 1: MEMORY ADDRESS TESTS
1825	T1 WRITE VALUE OF MEMORY ADDRESS INTO MEMORY
1869	T2 WRITE VALUE OF MEMORY ADDRESS INTO MEMORY
1909	T3 WRITE 1'S COMPLEMENT VALUE OF ADDRESS INTO ADDRESS.
1950	T4 WRITE BANK # INTO ALL ADDRESSES IN A 8K BANK
1985	T5 WRITE 1'S COMPLEMENT OF BANK #.
2041	SECTION 2: WORST CASE NOISE TESTS
2046	T6 WRITE A CONSTANT INTO MEMORY.
2065	T7 READ MEMORY AND COMPARE TO CONSTANT.
2097	T10 WORSE CASE NOISE (PARITY) WORD TESTING
2120	T11 ROTATE A '0' BIT THROUGH A FIELD OF ONES.
2142	T12 ROTATE A '1' BIT THROUGH A FIELD OF ZEROS
2164	T13 WORSE CASE NOISE PARITY BYTE TESTING
2362	T14 RANDOM DATA TESTING THRU PROGRAM CODE RELOCATION.
2397	SECTION 3: INSTRUCTION EXECUTION TESTS.
2398	T15 EXECUTE DATI, DATO THRU MEMORY.
2444	T16 EXECUTE DATI, DATOB (LOW BYTE) THRU MEMORY.
2490	T17 EXECUTE DATI, DATOB (HIGH BYTE) THRU MEMORY.
2537	T20 EXECUTE DATI, DATIO, DATO THRU MEMORY.
2585	T21 EXECUTE DATI, DATI, DATIO, DATOB (LOW BYTE) THRU MEMORY.
2631	T22 EXECUTE DATI, DATI, DATIO, DATOB (HIGH BYTE) THRU MEMORY.
2678	SECTION 4: MOS TESTS
2679	T23 MARCHING 1'S AND 0'S.
2768	T24 WRITE CHECKERBOARD STARTING WITH '125252' DATA.
2809	T25 WRITE CHECKERBOARD STARTING WITH 052525 DATA
2840	DONE: RELOCATE PROGRAM AND REPEAT ALL TESTS.
2870	END OF PASS ROUTINE
2919	SUBROUTINE AND TRAP ROUTINE SECTION.
2920	MEMORY MANAGEMENT AND ADDRESSING SUBROUTINES.
3174	SUBROUTINES FOR ADDRESS AND WORSE CASE NOISE TESTS.
3300	RELOCATION SUBROUTINES.
3502	PARITY MEMORY TRAP SERVICE AND SUBROUTINES.
3775	SUBROUTINES TO SET UP DATA FOR ERROR PRINTOUT ROUTINE.
3972	SCOPE HANDLER ROUTINE

4051	ERROR HANDLER ROUTINE
4120	ERROR MESSAGE TYPEOUT ROUTINE
4218	TTY INPUT ROUTINE
4416	READ AN OCTAL NUMBER FROM THE TTY
4493	TYPE ROUTINE
4591	APT COMMUNICATIONS ROUTINE
4686	CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
4754	BINARY TO OCTAL (ASCII) AND TYPE
4846	PHYSICAL ADDRESS TYPE ROUTINE
4905	STANDARD PROGRAM MESSAGES
5044	ERROR REPORTING MESSAGES AND TABLES.


```

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

```

13      .SBTTL OPERATIONAL SWITCH SETTINGS
14      .*
15      .*      SWITCH      USE
16      .*      -----
17      .*      15      HALT ON ERROR
18      .*      14      LOOP ON TEST
19      .*      13      INHIBIT ERROR TYPEOUTS
20      .*      12      INHIBIT KT11 (AT START TIME ONLY)
21      .*      11      INHIBIT ITERATIONS
22      .*      10      BELL ON ERROR
23      .*      9       LOOP ON ERROR
24      .*      8       LOOP ON TEST IN SWR<4:0>
25      .*      7       INHIBIT PROGRAM RELOCATION
26      .*      6       INHIBIT PARITY ERROR DETECTION
27      .*      5       INHIBIT EXERCISING VECTOR AREA.
28      .SBTTL BASIC DEFINITIONS
29
30      .*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
31      001100  STACK= 1100
32      .EQUIV EMT,ERROR      ;;BASIC DEFINITION OF ERROR CALL
33      .EQUIV IOT,SCOPE     ;;BASIC DEFINITION OF SCOPE CALL
34
35      .*MISCELLANEOUS DEFINITIONS
36      000011  HT= 11      ;;CODE FOR HORIZONTAL TAB
37      000012  LF= 12      ;;CODE FOR LINE FEED
38      000015  CR= 15      ;;CODE FOR CARRIAGE RETURN
39      000200  CRLF= 200    ;;CODE FOR CARRIAGE RETURN-LINE FEED
40      177776  PS= 177776  ;;PROCESSOR STATUS WORD
41      .EQUIV PS,PSW
42      177774  STKLMT= 177774  ;;STACK LIMIT REGISTER
43      177772  PIRQ= 177772  ;;PROGRAM INTERRUPT REQUEST REGISTER
44      177570  DSWR= 177570  ;;HARDWARE SWITCH REGISTER
45      177570  DDISP= 177570  ;;HARDWARE DISPLAY REGISTER
46
47      .*GENERAL PURPOSE REGISTER DEFINITIONS
48      000000  R0= %0      ;;GENERAL REGISTER
49      000001  R1= %1      ;;GENERAL REGISTER
50      000002  R2= %2      ;;GENERAL REGISTER
51      000003  R3= %3      ;;GENERAL REGISTER
52      000004  R4= %4      ;;GENERAL REGISTER
53      000005  R5= %5      ;;GENERAL REGISTER
54      000006  R6= %6      ;;GENERAL REGISTER
55      000007  R7= %7      ;;GENERAL REGISTER
56      000006  SP= %6      ;;STACK POINTER
```

```

57          000007          PC=      27          ;;PROGRAM COUNTER
58
59          ;*PRIORITY LEVEL DEFINITIONS
60          000000          PR0=      0          ;;PRIORITY LEVEL 0
61          000040          PR1=     40          ;;PRIORITY LEVEL 1
62          000100          PR2=    100          ;;PRIORITY LEVEL 2
63          000140          PR3=    140          ;;PRIORITY LEVEL 3
64          000200          PR4=    200          ;;PRIORITY LEVEL 4
65          000240          PR5=    240          ;;PRIORITY LEVEL 5
66          000300          PR6=    300          ;;PRIORITY LEVEL 6
67          000340          PR7=    340          ;;PRIORITY LEVEL 7
68
69          ;*'SWITCH REGISTER' SWITCH DEFINITIONS
70          100000          SW15=   100000
71          040000          SW14=    40000
72          020000          SW13=    20000
73          010000          SW12=    10000
74          004000          SW11=    4000
75          002000          SW10=    2000
76          001000          SW09=    1000
77          000400          SW08=    400
78          000200          SW07=    200
79          000100          SW06=    100
80          000040          SW05=    40
81          000020          SW04=    20
82          000010          SW03=    10
83          000004          SW02=    4
84          000002          SW01=    2
85          000001          SW00=    1
86          .EQUIV SW09,SW9
87          .EQUIV SW08,SW8
88          .EQUIV SW07,SW7
89          .EQUIV SW06,SW6
90          .EQUIV SW05,SW5
91          .EQUIV SW04,SW4
92          .EQUIV SW03,SW3
93          .EQUIV SW02,SW2
94          .EQUIV SW01,SW1
95          .EQUIV SW00,SW0
96
97          ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
98          100000          BIT15=  100000
99          040000          BIT14=   40000
100         020000          BIT13=   20000
101         010000          BIT12=   10000
102         004000          BIT11=   4000
103         002000          BIT10=   2000
104         001000          BIT09=   1000
105         000400          BIT08=   400
106         000200          BIT07=   200
107         000100          BIT06=   100
108         000040          BIT05=   40
109         000020          BIT04=   20
110         000010          BIT03=   10
111         000004          BIT02=    4
112         000002          BIT01=    2
    
```



```

113      000001      BIT00= 1
114      .EQUIV BIT09,BIT9
115      .EQUIV BIT08,BIT8
116      .EQUIV BIT07,BIT7
117      .EQUIV BIT06,BIT6
118      .EQUIV BIT05,BIT5
119      .EQUIV BIT04,BIT4
120      .EQUIV BIT03,BIT3
121      .EQUIV BIT02,BIT2
122      .EQUIV BIT01,BIT1
123      .EQUIV BIT00,BIT0
124
125      ;*BASIC "CPU" TRAP VECTOR ADDRESSES
126      000004      ERRVEC= 4      ;;TIME OUT AND OTHER ERRORS
127      000010      RESVEC= 10     ;;RESERVED AND ILLEGAL INSTRUCTIONS
128      000014      TBITVEC=14     ;;'T' BIT
129      000014      TRTVEC= 14     ;;TRACE TRAP
130      000014      BPTVEC= 14     ;;BREAKPOINT TRAP (BPT)
131      000020      IOTVEC= 20     ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
132      000024      PWRVEC= 24     ;;POWER FAIL
133      000030      EMTVEC= 30     ;;EMULATOR TRAP (EMT) **ERROR**
134      000034      TRAPVEC=34     ;;'TRAP' TRAP
135      000060      TKVEC= 60      ;;TTY KEYBOARD VECTOR
136      000064      TPVEC= 64     ;;TTY PRINTER VECTOR
137      000240      PIRQVEC=240   ;;PROGRAM INTERRUPT REQUEST VECTOR
138
139
140      .SBTTL MEMORY MANAGEMENT DEFINITIONS
141
142      ;*KT11 VECTOR ADDRESS
143
144      000250      MMVEC= 250
145
146      ;*KT11 STATUS REGISTER ADDRESSES
147
148      177572      SR0= 177572
149      177574      SR1= 177574
150      177576      SR2= 177576
151      172516      SR3= 172516
152
153      ;*KERNEL 'I' PAGE DESCRIPTOR REGISTERS
154
155      172300      KIPDR0= 172300
156      172302      KIPDR1= 172302
157      172304      KIPDR2= 172304
158      172306      KIPDR3= 172306
159      172310      KIPDR4= 172310
160      172312      KIPDR5= 172312
161      172314      KIPDR6= 172314
162      172316      KIPDR7= 172316
163
164      ;*KERNEL 'I' PAGE ADDRESS REGISTERS
165
166      172340      KIPAR0= 172340
167      172342      KIPAR1= 172342
168      172344      KIPAR2= 172344
  
```

```

169      172346      KIPAR3= 172346
170      172350      KIPAR4= 172350
171      172352      KIPAR5= 172352
172      172354      KIPAR6= 172354
173      172356      KIPAR7= 172356
174
175      000000      UP = 0                ;CODE FOR UPWARDS MAP IN MEM MGMT PDR'S
176      000006      RW = 6                ;CODE FOR READ/WRITE IN MEM MGMT PDR'S
177
178      ;* PARITY MEMORY DEFINITIONS.
179      000001      AE=1                    ;PARITY ACTION ENABLE
180      000114      PARVEC=114              ;PARITY TRAP VECTOR
181
182      ;* MISCELLANEOUS ASSIGNMENTS
183      017777      MASK4K= 17777          ;MASK FOR 4K ADDRESS BANK BOUNDARY.
184      037777      MASK8K= 37777        ;MASK FOR 8K ADDRESS BANK BOUNDARY
185      000007      MFPT= 7               ;MOVE FROM PROCESSOR TYPE TO R0
186
187      .SBTTL TRAP CATCHER
188
189
190      000000      .=0
191      ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
192      ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
193      ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
194      000174      .=174
195      000174      000000      DISPREG: .WORD 0          ;;SOFTWARE DISPLAY REGISTER
196      000176      000000      SWREG: .WORD 0           ;;SOFTWARE SWITCH REGISTER
197      .SBTTL STARTING ADDRESS(ES)
198      000200      000137      003742      JMP @#START ;;JUMP TO STARTING ADDRESS OF PROGRAM
199      000204      000167      000070      JMP RESTAR  ;;RESTART ADDRESS, USING PREVIOUS PARAMETERS.
200
201      000004      000004      .=ERRVEC
202      000004      023252      .WORD ERRTRP
203      000006      000000      .WORD 0
204
205      .SBTTL ACT11 HOOKS
206
207      ;*****
208      ;HOOKS REQUIRED BY ACT11
209      000010      000010      $SVPC=.                ;SAVE PC
210      000046      000046      .=46
211      000C46      012560      $ENDAD                ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
212      000052      000052      .=52
213      000052      040000      .WORD BIT14          ;;2)SET LOC.52 TO BIT14
214      000010      000010      .=$SVPC              ;; RESTORE PC
    
```



```

215      000100 000100 000340 000002  .=100      :IF BEVENT IS ENABLED
216      000100 000104 000340 000002  .WORD      104,340,2  :IGNORE ITS INTERRUPT,DO RTI
217
218
219      000300      .=300
220      *****
221      * THE FOLLOWING ROUTINES ARE LOCATED IN THE VECTOR AREA (0-1000) SO THAT
222      * THEY CAN BE PROTECTED BY SELECTING SW05 (SEE DOCUMENT FOR USE OF SW05).
223      *****
224      000300 005005  RESTAR: CLR      R5      ;CLEAR FLAG TO INDICATE RESTART.
225      000302 000401      BR      REST1      ;GO RESTORE PROGRAM BEFORE RESTARTING.
226      000304 010705  RESTOR: MOV      PC,      R5      ;PUT DATA INTO FLAG FOR RESTORE.
227      000306 012706 001070  REST1: MOV      #$CMTAG,SP ;SET UP THE STACK POINTER.
228      000312 005767 000216      TST      PRGMAP      ;CHECK IF THE MEMORY HAS BEEN MAPPED.
229      000316 001002      BNE      REST2      ;BR IF MEMORY MAPPED.
230      000320 000167 003432      JMP      STARTA      ;GO START,IF NOT MAPPED
231      000324 005767 000206  REST2: TST      MMAVA      ;CHECK IF MEM MGMT AVAILABLE.
232      000330 001454      BEQ      10$      ;BR IF NO MEM MGMT.
233      000332 032737 000001 177572  BIT      #BIT0, @#SR0 ;CHECK IF MEM MGMT ACTIVE.
234      000340 001027      BNE      2$      ;BR IF MEM MGMT ALREADY SET UP.
235      000342 012700 172300      MOV      #KIPDR0,R0 ;POINT TO FIRST MEM MGMT DDATA REG.
236      000346 012701 000010      MOV      #8,      R1      ;SET UP COUNTER.
237      000352 012720 077406 1$:      MOV      #077406,(R0)+ ;MAP FIRST 28K 1-FOR-1.
238      000356 005301      DEC      R1      ;COUNT REGISTERS.
239      000360 001374      BNE      1$      ;BR IF MORE REG.
240      000362 012700 172340      MOV      #KIPAR0,R0 ;POINT TO FIRST MEM MGMT ADDRESS REG.
241      000366 005020      CLR      (R0)+      ;PAR0 MAPPED INTO BANK0.
242      000370 012720 000200      MOV      #200, (R0)+ ;PAR1 MAPPED INTO BANK1.
243      000374 005020      CLR      (R0)+      ;PAR2 CLEARED
244      000376 005020      CLR      (R0)+      ;PAR3 CLEARED
245      000400 005020      CLR      (R0)+      ;PAR4 CLEARED
246      000402 005020      CLR      (R0)+      ;PAR5 CLEARED
247      000404 005020      CLR      (R0)+      ;PAR6 CLEARED
248      000406 012720 177600      MOV      #177600,(R0)+ ;PAR7 MAPPED INTO LAST BANK.
249      000412 012737 000001 177572  MOV      #BIT0, @#SR0 ;ENABLE MEM MGMT.
250      000420 005000 2$:      CLR      R0      ;INIT TEMP PAR REG.
251      000422 016701 000106      MOV      PRGMAP, R1 ;GET THE PROGRAM MAP...LO 64K.
252      000426 006201 3$:      ASR      R1      ;SHIFT THE MAP POINTER...
253      000430 103403      BCS      4$      ;BR WHEN FIRST 8K BANK FOUND.
254      000432 062700 000400      ADD      #400, R0 ;UPDATE TMP PAR TO NEXT 8K BANK.
255      000436 000773      BR      3$      ;NEXT 8K BANK
256      000440 010037 172340 4$:      MOV      R0, @#KIPAR0 ;PUT TEMP PAR INTO FIRST PAR.
257      000444 000137 000450      JMP      @#5$      ;JUMP INTO PROGRAM IF NOT THERE ALREADY.
258      000450 062700 000200 5$:      ADD      #200, R0 ;KEEP UPDATING TEMP PAR REG.
259      000454 010037 172342      MOV      R0, @#KIPAR1 ;SET UP SECOND PROGRAM BANK POINTER.
260      000460 000410      BR      20$      ;BR TO RELOCATE SECTION.
261      000462 016700 000044 10$:      MOV      RELOCF, R0 ;GET RELOCATION FACTOR.
262      000466 062700 001070      ADD      #$CMTAG,R0 ;SET UP STACK POINTER.
263      000472 010006      MOV      R0, SP ;SET STACK TO RELOCATE PROGRAM.
264      000474 062700 177412      ADD      #20$-#$CMTAG,R0 ;ADJUST R0 TO RELOCATED '20$' ADDRESS.
265      000500 000110      JMP      (R0) ;GO TO '20$' (RELOCATED).
266      000502 022767 000001 000024 20$:      CMP      #1, PRGMAP ;CHECK IF PROGRAM IS IN 8K BANK 0.
267      000510 001402      BEQ      21$      ;BR IF IN 8K BANK 0.
268      000512 004767 014370      JSR      PC, RELO ;RELOCATE THE PROGRAM BACK TO BANK 0.
269      000516 005067 000436 21$:      CLR      $TIMES ;CLEAN UP BEFORE STARTING.
270      000522 105067 000344      CLRB    $STNM
    
```

271 000526 000167 006130

JMP START1

;RESTART WITH PREVIOUSLY SELECTED PARAMETERS.

272
273
274
275
276
277
278
279
280

;* 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.

RELOCF: .WORD 0
PRGMAP: .WORD 0
MMAVA: .WORD 0

;CONTAINS RELOCATION FACTOR (NO MEM MGMT)
;PROGRAM MAP - WHERE THE PROGRAM IS LOCATED
;MEMORY MANAGEMENT AVAILABLE FLAG.
;BIT 0 = 1 MMU AVAILABLE
;BIT 15 = 1 22 BIT ADDRESSING AVAILABLE

.SBTTL POWER DOWN AND UP ROUTINES

```

281
282
283
284
285 000540 012737 000706 000024 $PWRDN: MOV #SILLUP,@#PWRVEC ;;SET FOR FAST UP
286 000546 012737 000340 000026 MOV #340,@#PWRVEC+2 ;;PRIO:7
287 000554 010046 MOV R0,-(SP) ;;PUSH R0 ON STACK
288 000556 010146 MOV R1,-(SP) ;;PUSH R1 ON STACK
289 000560 010246 MOV R2,-(SP) ;;PUSH R2 ON STACK
290 000562 010346 MOV R3,-(SP) ;;PUSH R3 ON STACK
291 000564 010446 MOV R4,-(SP) ;;PUSH R4 ON STACK
292 000566 010546 MOV R5,-(SP) ;;PUSH R5 ON STACK
293 000570 017746 000334 MOV @SWR,-(SP) ;;PUSH @SWR ON STACK
294 000574 010667 000112 MOV SP,$SAVR6 ;;SAVE SP
295 000600 012737 000612 000024 MOV #PWRUP,@#PWRVEC ;;SET UP VECTOR
296 000606 000000 HALT
297 000610 000776 BR -2 ;;HANG UP
298
299
300
301 000612 012737 000706 000024 $PWRUP: MOV #SILLUP,@#PWRVEC ;;SET FOR FAST DOWN
302 000620 016706 000066 MOV $SAVR6,SP ;;GET SP
303 000624 005067 000062 CLR $SAVR6 ;;WAIT LOOP FOR THE TTY
304 000630 005267 000056 1$: INC $SAVR6 ;;WAIT FOR THE INC
305 000634 001375 BNE 1$ ;;OF WORD
306 000636 012677 000266 MOV (SP)+,@SWR ;;POP STACK INTO @SWR
307 000642 012605 MOV (SP)+,R5 ;;POP STACK INTO R5
308 000644 012604 MOV (SP)+,R4 ;;POP STACK INTO R4
309 000646 012603 MOV (SP)+,R3 ;;POP STACK INTO R3
310 000650 012602 MOV (SP)+,R2 ;;POP STACK INTO R2
311 000652 012601 MOV (SP)+,R1 ;;POP STACK INTO R1
312 000654 012600 MOV (SP)+,R0 ;;POP STACK INTO R0
313 000656 012737 000540 000024 MOV #PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
314 000664 012737 000340 000026 MOV #340,@#PWRVEC+2 ;;PRIO:7
315 000672 004567 020676 JSR R5,$PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
316 000676 023761 $PWMSG: .WORD PWMSG ;;POWER FAIL MESSAGE POINTER
317 000700 012716 MOV (PC)+,(SP) ;;RESTART AT RESTART
318 000702 000300 $PWRAD: .WORD RESTART ;;RESTART ADDRESS
319 000704 000002 RTI
320 000706 000000 $SILLUP: HALT ;;THE POWER UP SEQUENCE WAS STARTED
321 000710 000776 BR -2 ;; BEFORE THE POWER DOWN WAS COMPLETE
322 000712 000000 $SAVR6: 0 ;;PUT THE SP HERE

```

323
 324
 325
 326
 327
 328
 329 001070
 330 001070 001070
 331 001070 000000
 332 001072 000
 333 001073 000
 334 001074 000000
 335 001076 000000
 336 001100 000000
 337 001102 000000
 338 001104 000
 339 001105 001
 340 001106 000000
 341 001110 000000
 342 001112 000000
 343 001114 000000
 344 001116 000000
 345 001120 000000
 346 001122 000000
 347 001124 000
 348 001125 000
 349 001126 000000
 350 001130 177570
 351 001132 177570
 352 001134 177560
 353 001136 177562
 354 001140 177564
 355 001142 177566
 356 001144 000
 357 001145 002
 358 001146 012
 359 001147 000
 360 001150 000000
 361 001152 000000
 362 001154 000000
 363 001156 000000
 364 001160 000000
 365 001162 000000
 366 001164 177607 000377
 367 001170 077
 368 001171 015
 369 001172 000012
 370
 371
 372
 373
 374
 375 001174
 376 001174 000000
 377 001176 000000
 378 001200 000000

.SBTTL COMMON TAGS

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

SCMTAG: .=1070

:::START OF COMMON TAGS

.WORD 0
 \$TSTN: .BYTE 0
 \$ERFLG: .BYTE 0
 \$ICNT: .WORD 0
 \$LPADR: .WORD 0
 \$LPERR: .WORD 0
 \$ERTTL: .WORD 0
 \$ITEMB: .BYTE 0
 \$ERMAX: .BYTE 1
 \$ERRPC: .WORD 0
 \$GDADR: .WORD 0
 \$BDADR: .WORD 0
 \$GDDAT: .WORD 0
 \$BDDAT: .WORD 0
 .WORD 0
 .WORD 0
 \$AUTOB: .BYTE 0
 \$INTAG: .BYTE 0
 .WORD 0
 SWR: .WORD DSWR
 DISPLAY: .WORD DDISP
 \$TKS: 177560
 \$TKB: 177562
 \$TPS: 177564
 \$TPB: 177566
 \$NULL: .BYTE 0
 \$FILLS: .BYTE 2
 \$FILLC: .BYTE 12
 \$TPFLG: .BYTE 0
 \$TMP0: .WORD 0
 \$TMP1: .WORD 0
 \$TMP2: .WORD 0
 \$TMP3: .WORD 0
 \$TIMES: 0
 \$ESCAPE: 0
 \$BELL: .ASCIZ <207><377><377>
 \$QUES: .ASCII /?/
 \$CRLF: .ASCII <15>
 \$LF: .ASCIZ <12>

:::CONTAINS THE TEST NUMBER
 :::CONTAINS ERROR FLAG
 :::CONTAINS SUBTEST ITERATION COUNT
 :::CONTAINS SCOPE LOOP ADDRESS
 :::CONTAINS SCOPE RETURN FOR ERRORS
 :::CONTAINS TOTAL ERRORS DETECTED
 :::CONTAINS ITEM CONTROL BYTE
 :::CONTAINS MAX. ERRORS PER TEST
 :::CONTAINS PC OF LAST ERROR INSTRUCTION
 :::CONTAINS ADDRESS OF 'GOOD' DATA
 :::CONTAINS ADDRESS OF 'BAD' DATA
 :::CONTAINS 'GOOD' DATA
 :::CONTAINS 'BAD' DATA
 :::RESERVED--NOT TO BE USED
 :::AUTOMATIC MODE INDICATOR
 :::INTERRUPT MODE INDICATOR
 :::ADDRESS OF SWITCH REGISTER
 :::ADDRESS OF DISPLAY REGISTER
 :::TTY KBD STATUS
 :::TTY KBD BUFFER
 :::TTY PRINTER STATUS REG. ADDRESS
 :::TTY PRINTER BUFFER REG. ADDRESS
 :::CONTAINS NULL CHARACTER FOR FILLS
 :::CONTAINS # OF FILLER CHARACTERS REQUIRED
 :::INSERT FILL CHARS. AFTER A 'LINE FEED'
 :::'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)
 :::USER DEFINED
 :::USER DEFINED
 :::USER DEFINED
 :::USER DEFINED
 :::MAX. NUMBER OF ITERATIONS
 :::ESCAPE ON ERROR ADDRESS
 :::CODE FOR BELL
 :::QUESTION MARK
 :::CARRIAGE RETURN
 :::LINE FEED

.SBTTL APT MAILBOX-ETABLE

 .EVEN
 \$MAIL: :::APT MAILBOX
 \$MSGTY: .WORD AMSGTY :::MESSAGE TYPE CODE
 \$FATAL: .WORD AFATAL :::FATAL ERROR NUMBER
 \$TESTN: .WORD ATESTN :::TEST NUMBER

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

491
 492
 493
 494 001504 000000
 495 001506 070032
 496
 497 001510 000000
 498 001512 000000
 499
 500 001514
 501
 502 001514 000000
 503 001516 000000
 504 001520 000000
 505 001522 000000
 506 001524 000000
 507 001526 000000
 508 001530 000000
 509 001532 000000
 510 001534 000000
 511 001536 000000
 512 001540 000000
 513 001542 000000
 514 001544 000000
 515 001546 000000
 516 001550 000000
 517 001552 000000
 518
 519 001554
 520
 521 001554 000000
 522 001556 000000
 523 001560 000000
 524 001562 000000
 525 001564 000000
 526 001566 000000
 527 001570 000000
 528 001572 000000
 529 001574 000000
 530 001576 000000
 531 001600 000000
 532 001602 000000
 533 001604 000000
 534 001606 000000
 535 001610 000000
 536 001612 000000
 537
 538 001614
 539
 540
 541 001614 000000
 542 001616 000000
 543 001620 000000
 544 001622 000000
 545 001624 000000
 546 001626 000000

```

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

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

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

SAVTST: ;SAVED TEST MAP - USED DURING FIRST PASS TO ONLY
        ; TEST EACH BANK ONCE.
        .WORD 0 ;1ST WORD MAP...128KW
        .WORD 0 ;2ND WORD MAP...256KW
        .WORD 0 ;3RD WORD MAP...384KW
        .WORD 0 ;4TH WORD MAP...512KW
        .WORD 0 ;5TH WORD MAP...640KW
        .WORD 0 ;6TH WORD MAP...768KW
    
```

547	001630	000000	.WORD	0	:7TH WORD MAP...896KW
548	001632	000000	.WORD	0	:8TH WORD MAP...1024KW
549	001634	000000	.WORD	0	:9TH WORD MAP...1152KW
550	001636	000000	.WORD	0	:10TH WORD MAP...1280KW
551	001640	000000	.WORD	0	:11TH WORD MAP...1408KW
552	001642	000000	.WORD	0	:12TH WORD MAP...1536KW
553	001644	000000	.WORD	0	:13TH WORD MAP...1664KW
554	001646	000000	.WORD	0	:14TH WORD MAP...1792KW
555	001650	000000	.WORD	0	:15TH WORD MAP...1920KW
556	001652	000000	.WORD	0	:16TH WORD MAP...2048KW
557					
558	001654		PMEMAP:		:PARITY MAP - WHICH BANKS HAVE MEMORY PARITY
559					
560	001654	000000	.WORD	0	:1ST WORD MAP...128KW
561	001656	000000	.WORD	0	:2ND WORD MAP...256KW
562	001660	000000	.WORD	0	:3RD WORD MAP...384KW
563	001662	000000	.WORD	0	:4TH WORD MAP...512KW
564	001664	000000	.WORD	0	:5TH WORD MAP...640KW
565	001666	000000	.WORD	0	:6TH WORD MAP...768KW
566	001670	000000	.WORD	0	:7TH WORD MAP...896KW
567	001672	000000	.WORD	0	:8TH WORD MAP...1024KW
568	001674	000000	.WORD	0	:9TH WORD MAP...1152KW
569	001676	000000	.WORD	0	:10TH WORD MAP...1280KW
570	001700	000000	.WORD	0	:11TH WORD MAP...1408KW
571	001702	000000	.WORD	0	:12TH WORD MAP...1536KW
572	001704	000000	.WORD	0	:13TH WORD MAP...1664KW
573	001706	000000	.WORD	0	:14TH WORD MAP...1792KW
574	001710	000000	.WORD	0	:15TH WORD MAP...1920KW
575	001712	000000	.WORD	0	:16TH WORD MAP...2048KW
576					
577	001714		BITPT:		:POINTER TO CURRENT 8K BANK OF MEMORY
578					
579	001714	000000	.WORD	0	:1ST WORD MAP...128KW
580	001716	000000	.WORD	0	:2ND WORD MAP...256KW
581	001720	000000	.WORD	0	:3RD WORD MAP...384KW
582	001722	000000	.WORD	0	:4TH WORD MAP...512KW
583	001724	000000	.WORD	0	:5TH WORD MAP...640KW
584	001726	000000	.WORD	0	:6TH WORD MAP...768KW
585	001730	000000	.WORD	0	:7TH WORD MAP...896KW
586	001732	000000	.WORD	0	:8TH WORD MAP...1024KW
587	001734	000000	.WORD	0	:9TH WORD MAP...1152KW
588	001736	000000	.WORD	0	:10TH WORD MAP...1280KW
589	001740	000000	.WORD	0	:11TH WORD MAP...1408KW
590	001742	000000	.WORD	0	:12TH WORD MAP...1536KW
591	001744	000000	.WORD	0	:13TH WORD MAP...1664KW
592	001746	000000	.WORD	0	:14TH WORD MAP...1792KW
593	001750	000000	.WORD	0	:15TH WORD MAP...1920KW
594	001752	000000	.WORD	0	:16TH WORD MAP...2048KW
595					
596					
597					
598	001754	000000	MMORE: .WORD	0	:LOOP ADDRESS FOR MULTIPLE BLOCK TESTING.
599					:SET UP BY 'INITMM' AND 'INITDN' ROUTINES.
600					:USED BY 'MMUP' AND 'MMDOWN' ROUTINES.
601	001756	000	SELFLG: .BYTE	0	:OPERATOR SELECTED PARAMETERS FLAG. (SA=204)
602	001757	000	FLAG8K: .BYTE	0	:8K BLOCK INDICATOR. USED IN 'INITMM' AND 'MMUP'.

603 001760 000
 604 001762 001762
 605 001762 000000
 606
 607
 608 001764 000000
 609 001766 000004
 610 001770 000000
 611 001772 000000
 612 001774 000000
 613 001776 000000
 614 002000 000000
 615
 616
 617
 618
 619
 620
 621
 622 002002
 623 002002 001070
 624 002004 001514
 625 002006 001614
 626 002010 001554
 627 002012 001654
 628 002014 001714
 629 002016 001506
 630 002020 002270
 631 002022 003370
 632 002024 010356
 633 002026 002242
 634 002030 015454
 635 002032 003432
 636 002034 000010
 637 002036 012344
 638
 639
 640
 641 002040 001106 001110 001114
 642 002046 001116 000000
 643 002052 001504 001106 001110
 644 002060 001114 001116 000000
 645 002066 001504 001106 001110
 646 002074 001114 000000
 647 002100 001504 001106 001150
 648 002106 001110 000000
 649 002112 001504 001106 001110
 650 002120 001150 001114 001116
 651 002126 000000
 652 002130 001504 001106 001150
 653 002136 001110 001114 001116
 654 002144 000000
 655 002146 001504 001106 001110
 656 002154 001112 001114 001116
 657 002162 000000
 658 002164 001504 001106 001112

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

 * RELATIVE ADDRESSING TABLE.
 * THE FOLLOWING LOCATIONS ARE MODIFIED AT RELOCATION TIME TO ALLOW
 * RELATIVE ADDRESSING TO GET THE RELOCATED VALUE OF THE ARGUMENT TAGS.

RADTAB:
 .STACK: \$CMTAG ; STACK POINTER INITIAL ADDRESS.
 .MEMMAP: MEMMAP ; AUTO. MEMORY SIZING MAP
 .SAVTST: SAVTST ; SIZED MEMORY MAP FROM AUTO. OR USER
 .TSTMAP: TSTMAP ; MEMORY MAP OF 8K BANKS TO TEST
 .PMEAP: PMEAP ; PARITY MAP OF 8K BANKS
 .BITPT: BITPT ; BIT POINTER FOR BLOCK UNDER TEST
 .RESRV: RESRVD ; PARITY REGISTER RESERVED BIT MASK ADDRESS.
 .MPRO: MPRO ; MEMORY PARITY REGISTER TABLE ADDRESS.
 .MPRX: MPRX ; MEMORY PARITY REGISTER EXIST TABLE ADDRESS.
 .PBTRP: PBTRP ; PARITY BYTE TEST TRAP ROUTINE ADDRESS.
 .MPPAT: MPPATS ; MEMORY PARITY PATTERN TABLE ADDRESS.
 .PESRV: PESRV ; MEMORY PARITY ERROR TRAP ROUTINE ADDRESS.
 .ERRTB: \$ERRTB ; ERROR TYPEOUT TABLE PONTER.
 .EIGHT: 8. ; DECIMAL TYPE ROUTINE COUNT DESIGNATOR.
 .TST32: TST32 ; SCOPE ABORT ADR FOR WHEN NO MEM AVA FOR TEST.

 * DATA CONTAINERS FOR ERROR PRINTOUT.

DT1: \$ERRPC,\$GDADR,\$GDDAT,\$BDDAT,0
 DT2: \$VERPC,\$ERRPC,\$GDADR,\$GDDAT,\$BDDAT,0
 DT12: \$VERPC,\$ERRPC,\$GDADR,\$GDDAT,0
 DT14: \$VERPC,\$ERRPC,\$TMPO,\$GDADR,0
 DT15: \$VERPC,\$ERRPC,\$GDADR,\$TMPO,\$GDDAT,\$BDDAT,0
 DT21: \$VERPC,\$ERRPC,\$TMPO,\$GDADR,\$GDDAT,\$BDDAT,0
 DT23: \$VERPC,\$ERRPC,\$GDADR,\$BDADR,\$GDDAT,\$BDDAT,0
 DT24: \$VERPC,\$ERRPC,\$BDADR,0

```

659 002172 000000
660 002174 001504 001106 001112 DT25: $VERPC,$ERRPC,$BDADR,$TMP0,$TMP1,0
661 002202 001150 001152 000000
662 002210 001504 001106 001150 DT26: $VERPC,$ERRPC,$TMP0,$TMP1,0
663 002216 001152 000000
664 002222 001150 001152 001110 DT30: $TMP0,$TMP1,$GDADR,$BDDAT,0
665 002230 001116 000000
666 002234 001156 000000 DT31: $TMP3,0
667 002240 177777 .WORD -1 ;TABLE TERMINATOR.
    
```

```

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

```

674 002242 125325 MPPATS: 125325 ;EVEN,ODD
675 002244 152652 152652 ;ODD,EVEN
676 002246 052452 052452 ;EVEN,ODD
677 002250 025125 025125 ;ODD,EVEN
678 002252 102070 102070 ;EVEN,EVEN
679 002254 072527 072527 ;ODD,ODD
680 002256 177777 177777 ;EVEN,EVEN
681 002260 107030 107030 ;ODD,ODD
682 002262 152525 152525 ;ODD,EVEN
683 002264 000000 0 ;EXTRA PATTERN HOLDER FOR
684 ;FUTURE USE
685 002266 000000 MPEND: 0 ;TABLE TERMINATOR
    
```

```

.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
://////
    
```

```

700 002270 172101 MPRO: 172100 +1 ;PARITY STATUS REGISTER
701 002272 000000 .WORD 0 ;1ST WORD CONTROL MAP...128KW
702 002274 000000 .WORD 0 ;2ND WORD CONTROL MAP...256KW
703 002276 000000 .WORD 0 ;3RD WORD CONTROL MAP...384KW
704 002300 000000 .WORD 0 ;4TH WORD CONTROL MAP...512KW
705 002302 000000 .WORD 0 ;5TH WORD CONTROL MAP...640KW
706 002304 000000 .WORD 0 ;6TH WORD CONTROL MAP...768KW
707 002306 000000 .WORD 0 ;7TH WORD CONTROL MAP...896KW
708 002310 000000 .WORD 0 ;8TH WORD CONTROL MAP..1024KW
709 002312 000000 .WORD 0 ;9TH WORD CONTROL MAP..1152KW
710 002314 000000 .WORD 0 ;10TH WORD CONTROL MAP..1280KW
711 002316 000000 .WORD 0 ;11TH WORD CONTROL MAP..1408KW
712 002320 000000 .WORD 0 ;12TH WORD CONTROL MAP..1536KW
713 002322 000000 .WORD 0 ;13TH WORD CONTROL MAP..1664KW
714 002324 000000 .WORD 0 ;14TH WORD CONTROL MAP..1792KW
    
```


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

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

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

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

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

:THIS IS THE END OF THE TABLE !
 MPRX: .BLKW 17.
 :TABLE TO HOLD JUST PARITY STATUS REGISTERS THAT EXIST.
 : (THE EXTRA WORD IS FOR A TERMINATOR.)

988
 989 003370 000021
 990
 991

992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006 003432
1007 003432
1008
1009 003432 024774
1010 003434 026334
1011 003436 002040
1012 003440 027072
1013
1014 003442 025030
1015 003444 026357
1016
1017 003446 002052
1018 003450 027076
1019
1020 003452 025030
1021 003454 027036
1022 003456 002052
1023 003460 027103
1024
1025 003462 025064
1026 003464 026357
1027 003466 002052
1028 003470 027076
1029
1030 003472 025122
1031 003474 026357
1032 003476 002052
1033 003500 027076
1034
1035 003502 025160
1036 003504 026357
1037 003506 002052
1038 003510 027076
1039
1040 003512 025224
1041 003514 000000
1042 003516 000000
1043 003520 000000
1044
1045 003522 025245
1046 003524 026357
1047 003526 002052

.SBITL 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

\$ERRTB:
CHGG1:

;* ITEM 1
DM1 :PARITY REGISTER DATA ERROR.
DH1 :PC,REG,S/B,WAS
DT1 :\$ERRPC,\$GDADR,\$GDDAT,\$BDDAT
DF1 :16,22,16,16
;* ITEM 2
DM2 :ADDRESS TEST ERROR(TST1-5).
DH2 :V/PC,P/PC,MA,S/B,WAS
DT2 :\$VERPC,\$ERRPC,\$GDADR,\$GDDAT,\$BDDAT
DF2 :16,22,22,16,16
;* ITEM 3
DM2 :ADDRESS TEST ERROR(TST1-5).
DH31 :V/PC,P/PC,MA,S/B,WAS
DT2 :\$VERPC,\$ERRPC,\$GDADR,\$GDDAT,\$BDDAT
DF3 :16,22,22,8,8
;* ITEM 4
DM4 :CONSTANT DATA ERROR(TST6-10).
DH2 :V/PC,P/PC,MA,S/B,WAS
DT2 :\$VERPC,\$ERRPC,\$GDADR,\$GDDAT,\$BDDAT
DF2 :16,22,22,16,16
;* ITEM 5
DM5 :ROTATING BIT ERROR(TST11-12).
DH2 :V/PC,P/PC,MA,S/B,WAS
DT2 :\$VERPC,\$ERRPC,\$GDADR,\$GDDAT,\$BDDAT
DF2 :16,22,22,16,16
;* ITEM 6
DM6 :MOS REFRESH TEST ERROR (TST24-25).
DH2 :V/PC,P/PC,MA,S/B,WAS
DT2 :\$VERPC,\$ERRPC,\$GDADR,\$GDDAT,\$BDDAT
DF2 :16,22,22,16,16
;* ITEM 7
DM7 :FATAL ERROR HALT
0
0
0
;* ITEM 10
DM10 :MARCHING 1'S AND 0'S ERROR(TST23).
DH2 :V/PC,P/PC,MA,S/B,WAS
DT2 :\$VERPC,\$ERRPC,\$GDADR,\$GDDAT,\$BDDAT

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


```

1160      ::EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
1161 004020 013746 000004      MOV @#ERRVEC, -(SP)      ::SAVE ERROR VECTOR
1162 004024 012737 004060 000004      MOV #64$, @#ERRVEC      ::SET UP ERROR VECTOR
1163 004032 012767 177570 175070      MOV #DSWR, SWR          ::SETUP FOR A HARDWARE SWICH REGISTER
1164 004040 012767 177570 175064      MOV #DDISP, DISPLAY     ::AND A HARDWARE DISPLAY REGISTER
1165 004046 022777 177777 175054      CMP #-1, @SWR           ::TRY TO REFERENCE HARDWARE SWR
1166 004054 001012      BNE 66$                 ::BRANCH IF NO TIMEOUT TRAP OCCURRED
1167      BR 65$               ::AND THE HARDWARE SWR IS NOT = -1
1168 004056 000403      BR 65$                 ::BRANCH IF NO TIMEOUT
1169 004060 012716 004066 64$:      MOV #65$, (SP)         ::SET UP FOR TRAP RETURN
1170 004064 000002      RTI
1171 004066 012767 000176 175034 65$:      MOV #SWREG, SWR        ::POINT TO SOFTWARE SWR
1172 004074 012767 000174 175030      MOV #DISPREG, DISPLAY
1173 004102 012637 000004 66$:      MOV (SP)+, @#ERRVEC   ::RESTORE ERROR VECTOR
1174
1175 004106 005067 175070      CLR $PASS              ::CLEAR PASS COUNT
1176 004112 132767 000200 175075      BITB #APTSIZE, $ENVM   ::TEST USER SIZE UNDER APT
1177 004120 001403      BEQ 67$                ::YES, USE NON-APT SWITCH
1178 004122 012767 001216 175000      MOV #SSWREG, SWR      ::NO, USE APT SWITCH REGISTER
1179 004130 67$:
1180 004130 005067 175356      CLR LDDISP             ;CLEAR DISPLAY REGISTER STORAGE LOCN
1181 004134 005077 174772      CLR @DISPLAY           ;CLEAR DISPLAY REGISTER
1182      .SBTTL TYPE PROGRAM NAME
1183      ::TYPE THE NAME OF THE PROGRAM IF FIRST PASS
1184 004140 005227 177777      INC #-1                ::FIRST TIME?
1185 004144 001041      BNE 68$                ::BRANCH IF NO
1186 004146 022737 012560 000042      CMP #SENDAD, @#42     ::ACT-11?
1187 004154 001435      BEQ 68$                ::BRANCH IF YES
1188 004156 004567 015412      JSR R5, $PRINT         ;GO PRINT OUT THE FOLLOWING MESSAGE.
1189 004162 004236      .WORD 69$             ;ADDRESS OF MESSAGE TO BE TYPED
1190      .SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
1191 004164 005737 000042      TST @#42              ::ARE WE RUNNING UNDER XXDP/ACT?
1192 004170 001016      BNE 70$                ::BRANCH IF YES
1193 004172 126727 175016 000001      CMPB $ENV, #1         ::ARE WE RUNNING UNDER APT?
1194 004200 001412      BEQ 70$                ::BRANCH IF YES
1195 004202 026727 174722 000176      CMP SWR, #SWREG       ::SOFTWARE SWITCH REG SELECTED?
1196 004210 001011      BNE 71$                ::BRANCH IF NO
1197      ;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $GTSWR ROUTINE
1198      ;* WIHTOUT USING A 'TRAP' INSTRUCTION AS CALLED FOR BY **SYSMAC**
1199 004212 106746      MFPS -(SP)            ;PUT THE PROCESSOR STATUS ON THE STACK
1200 004214 105066 000001      CLR 1(SP)             ;HIGH BYTE CLEARED TO INSURE KERNEL MODE
1201      ;ON PSW RETURN.
1202 004220 004767 014256      JSR PC, $GTSWR        ;GO TO THE SUBROUTINE
1203 004224 000403      BR 71$
1204 004226 112767 000001 174670 70$:      MOV #1, $AUTOB        ::SET AUTO-MODE INDICATOR
1205 004234 71$:
1206 004234 000405      BR 68$                ::GET OVER THE ASCIZ
1207      ::69$: .ASCIZ <CRLF>'CVMSAA'<CRLF>
1208 68$:
1209 004250      MOV PC, R0            ;GET CURRENT PROGRAM COUNTER.
1210 004252 010700 004252 9$:      CMP #9$, R0           ;CHECK IF THE PROGRAM IS RELOCATED.
1211 004256 001402      BEQ 10$               ;BR IF PROGRAM NOT RELOCATED.
1212 004260 000167 174014      JMP RESTAR            ;GO TRY TO RELOCTED BEFORE CONTINUING.
1213 004264 012767 000001 174242 10$:      MOV #1, PRGMAP        ;INITIALIZE PROGRAM MAP.
1214 004272 005067 174234 12$:      CLR RELOCF           ;INIT THE RELOCATION FACTOR.
1215 004276 105737 001214      TSTB @#$ENV          ;CHECK FOR APT11

```

```

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

```


1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283 004522 012706 001100
1284 004526 012700 001514
1285 004532 005002
1286 004534 012703 000001
1287 004540 012737 004562 000004
1288 004546 011222
1289 004550 032702 037777
1290 004554 001374
1291 004556 050310
1292 004560 000420
1293
1294
1295 004562 062706 000004
1296 004566 022702 160000
1297 004572 001001
1298 004574 000405
1299 004576 022702 170000
1300 004602 001004
1301 004604 005267 175164
1302 004610 050310
1303 004612 000407
1304 004614 052702 037777
1305 004620 005202
1306
1307 004622 106303
1308 004624 032703 000020
1309 004630 001746
1310 004632 000500
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327 004634 012706 001100

```
*****  
* 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
```

```
*****  
* THIS ROUTINE WILL MAP MEMORY IN 8K SEGMENTS. MEMORY MANAGEMENT REGISTERS  
* KIPAR2 AND KIPAR3 ARE USED TO MAP THE 8K BANKS OF MEMORY.  
* IF MEMORY EXISTS NEXT TO THE I/O PAGE (I.E. 760000 OR 17760000)  
* THEN THE LAST BANK WILL BE ACKNOWLEDGED AS EXISTING.  
* STORAGE USED:  
* R0 = MEMMAP POINTER  
* R2 = ADDRESS POINTER  
* R3 = BANK POINTER  
* KIPAR2 = MAPPED TO 1ST 4K OF PRESENT 8K BANK  
* KIPAR3 = MAPPED TO 2ND 4K OF PRESENT 8K BANK  
* LITERALS USED:  
* MASK8K = MASK OF 8K (37777)  
*****
```

```
KTSIZ: MOV #STACK,SP ;SET-UP THE STACK
```

```

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



```

1384 005034 012700 001514 DISMAP: MOV #MEMMAP,R0 ;LOAD R0 WITH MEMMAP ADR
1385 005040 012737 023252 000004 MOV #ERRTRP,@#ERRVEC ;SET UP TIME OUT VECTOR
1386 005046 004567 014522 JSR R5,$SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
1387 005052 023564 .WORD MEMMES ;ADDRESS OF MESSAGE TO BE TYPED
1388 ;'MEMORY MAP:'
1389 005054 004767 011570 JSR PC,TYPMAP ;GO TYPE THE MAP
1390 005060 004567 014510 JSR R5,$SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
1391 005064 001171 .WORD $CRLF ;ADDRESS OF MESSAGE TO BE TYPED
1392 005066 012702 001614 MOV #SAVTST,R2 ;LOAD ADR OF SAVTST TABLE TO BE CLEARED
1393 005072 012700 001514 MOV #MEMMAP,R0 ;LOAD ADR OF MEMMAP TABLE
1394 005076 012701 000020 MOV #16.,R1 ;LOAD COUNTER
1395 005102 005012 1$: CLR (R2) ;CLEAR SAVTST TABLE ENTRY
1396 005104 012022 MOV (R0)+,(R2)+ ;LOAD SAVTST FROM MEMMAP
1397 005106 077103 SOB R1,1$ ;DECREMENT CTR 16 TIMES
1398 005110 016700 174400 MOV MEMMAP,R0 ;LOAD R0 WITH MAP OF 1ST 128K
1399 005114 042700 177774 BIC #177774,R0 ;MASK ALL BUT BOTTOM 16K
1400 005120 022700 000003 CMP #3,R0 ;IF BOTTOM 16K IS ALL THERE
1401 005124 001404 BEQ GMPR ; THEN GO RUN
1402 005126 004567 014442 JSR R5,$SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
1403 005132 023644 .WORD INSUFF ;ADDRESS OF MESSAGE TO BE TYPED
1404 ;'FIRST 16K OF MEMORY NOT ALL THERE!''
1405 005134 000000 HALT ;FATAL ERROR HALT
1406 ;MEMORY IS NOT CONFIGURED TO RUN THIS PROGRAM
1407
1408
1409
1410
1411
1412
1413
1414

```

.SBTTL MAP PARITY REGISTERS

```

*****
;* SEARCH FOR PARITY REGISTERS PRESENT AND TYPE ADDRESSES OF THOSE FOUND
;* THAT ARE FUNCTIONAL AND HAVE CORRESPONDING PARITY MEMORY
*****

```

```

1415 005136 012704 003370 GMPR: MOV #MPRX,R4 ;SET UP POINTER TO PARITY REG EXIST TABLE.
1416 005142 032777 000100 173760 BIT #SW06,@SWR ;CHECK FOR INHIBIT PARITY SWITCH.
1417 005150 001040 BNE GMPRD ;BR IF INHIBIT PARITY.
1418 005152 012703 002270 MOV #MPRO,R3 ;SET UP TABLE POINTER
1419 005156 012737 005200 000004 GMPRA: MOV #GMPRB,@#ERRVEC ;SET UP TIMEOUT TRAP SERVICE
1420 005164 042713 000001 BIC #1,(R3) ;CLEAR FLAG BIT IN TABLE
1421 005170 005773 000000 TST @(R3) ;DOES THIS MEMORY PARITY REGISTER EXIST.
1422 ;* IF IT DOESN'T EXIST, A TIMEOUT TRAP WILL GO TO 'GMPRB'.
1423 005174 012324 MOV (R3)+,(R4)+ ;SAVE IT IN THE PARITY REG EXIST TABLE.
1424 005176 000404 BR GMPRBA ;SKIP TIMEOUT SERVICE CODE
1425 ;* TIMEOUT COMES HERE
1426 005200 062706 000004 GMPRB: ADD #4,SP ;RESTORE STACK POINTER
1427 005204 052723 000001 BIS #1,(R3)+ ;SET FLAG TO INDICATE REGISTER NOT PRESENT
1428 005210 012701 000021 GMPRBA: MOV #17.,R1 ;LOAD COUNTER
1429 005214 005023 GMPRC: CLR (R3)+ ;CLEAR ENTRY IN THIS GPR TABLE
1430 005216 077102 SOB R1,GMPRC ;DECREMENT COUNTER AND CONTINUE
1431 ;CLEARING COUNTER UNTIL COMPLETED.
1432 005220 020327 003370 CMP R3,#MPRX ;HAVE WE CHECKED ALL REGISTERS?
1433 ;NO - GO BACK TO CHECK NEXT ONE
1434 005224 103757 BLO GMPRA ;SET TERMINATOR IN PARITY REG EXIST TABLE.
1435 005226 005014 CLR (R4)
1436 005230 012737 023252 000004 MOV #ERRTRP,@#ERRVEC ;RESTORE TRAPCATCHER
1437 005236 005767 176126 TST MPRX ;ANY PARITY REGISTERS PRESENT?
1438 005242 001006 BNE MPAMEM ;YES - GO TEST CONTROLS PRESENT
1439 005244 004567 014324 JSR R5,$SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.

```

1440 005250 023725
1441
1442 005252 005014
1443 005254 000167 000744
1444

.WORD MTR
GMPRD: CLR (R4)
JMP MANUAL

:ADDRESS OF MESSAGE TO BE TYPED
: 'NO MEMORY PARITY REGISTERS FOUND'
: MAKE SURE TABLE IS CLEAR.
: AND SKIP ALL CONTROLS TESTING

1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500

005260 004767 011230
005264 012702 014000
005270 005767 173242
005274 001404
005276 012702 054000
005302 004767 005330

005306 012703 001654
005312 012704 000020
005316 005023
005320 077402

005322 004767 011166
005326 005000
005330 012705 000002
005334 012701 000001

005340 012703 002270
005344 010304
005346 060504

005350 032713 000001
005354 001021
005356 013773 001766 000000
005364 011212
005366 005712
005370 043773 001766 000000
005376 005773 000000
005402 100006
005404 012763 070032 000042
005412 050160 001654
005416 050114
005420 062703 000044
005424 022703 003370

```
.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
*****
MPAMEM: JSR PC,CLRPAR ;INITIALIZE ALL PARITY REGISTERS
        MOV #14000,R2 ;SET ADDRESS TO 14000 TO WRITE WRONG PARITY
        TST MMAVA ;IF NO MEM MGMT
        BEQ MAPRB ; THEN GO MAP PARITY MEMORY
        MOV #54000,R2 ; ELSE SET ADDRESS POINTER TO MAP THRU PAR2
        JSR PC,MMINIT ;SET UP MEM MGMT REGISTERS

MAPRB: MOV #PMEMAP,R3 ;LOAD PMEMAP TABLE ADR
        MOV #16,R4 ;LOAD COUNTER
1$: CLR (R3)+ ;CLEAR ALL OF TABLE
     SOB R4,1$ ;IF NOT DONE CLEARING THEN TRY SOMEMORE

        JSR PC,CLRPAR ;GO INITIALIZE ALL PARITY REGISTERS
        CLR R0 ;INIT INDEX FOR MEMMAP AND PMEMAP TABLES
        MOV #2,R5 ;INIT INDEX FOR MPR TABLES
        MOV #BIT0,R1 ;INIT BANK POINTER

2$: MOV #MPR0,R3 ;INIT MPR TABLE ADDRESS POINTER
3$: MOV R3,R4 ;UPDATE TO NEW TABLE
     ADD R5,R4 ;UPDATE INDEX THRU NEW TABLE

        BIT #BIT0,(R3) ;IF CSR IS NOT PRESENT
        BNE 4$ ; THEN GO TRY AGAIN
        MOV @#WWP,@(R3) ; ELSE SET WRITE WRONG PARITY BIT
        MOV (R2),(R2) ;WRITE WRONG PARITY
        TST (R2) ;READ WRONG PARITY
        BIC @#WWP,@(R3) ;CLEAR WRITE WRONG PARITY BIT
        TST @(R3) ;IF NO PARITY ERROR
        BPL 4$ ; THEN REGISTER DOES NOT CONTROL THIS MEMORY
        MOV #70032,42(R3) ; ELSE SAVE THE PARITY MASK IN TABLE
        BIS R1,PMEMAP(R0) ;SET BANK IN PMEMAP
        BIS R1,(R4) ;SET BANK IN MPR TABLES
4$: ADD #4,R3 ;UPDATE TO NEXT MPR TABLE
     CMP #MPRX,R3 ; IF NOT END OF TABLE
```

```

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



```

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

```

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

1669	006150	022702	003370	2\$:	CMP	#MPRX,R2		:IF END OF TABLE
1670	006154	003423			BLE	MANUAL		: THEN BRANCH
1671	006156	032712	000001		BIT	#BIT0,(R2)		: ELSE IF CSR DOES NOT EXIST
1672	006162	001015			BNE	65\$: THEN BRANCH
1673	006164	017201	000000		MOV	@(R2),R1		: ELSE SAVE CONTENTS OF CSR
1674	006170	005072	000000		CLR	@(R2)		: CLEAR THE CSR
1675	006174	042701	037772		BIC	#37772,R1		: TEST FOR BIT14 TOO
1676								: 18 BIT MODE = BIT14 ALWAYS READ AS ZERO
1677								: 22 BIT MODE = BIT14 R/W,CLEARED BY RESET
1678	006200	005701		4\$:	TST	R1		: CHECK IF REST WERE CLEARED BY RESET
1679	006202	001405			BEQ	65\$: BRANCH OVER ERROR CALL IF GOOD DATA.
1680	006204	004767	010326	64\$:	JSR	PC,	SPRNT	: SET UP VALUES FOR ERROR PRINTING.
1681	006210	004767	011452		JSR	PC,	\$ERROR	: *** ERROR *** (GO TYPE A MESSAGE)
1682	006214	000001			.WORD	1		: ERROR TYPE CODE.
1683	006216			65\$:				
1684	006216	062702	000044		ADD	#44,R2		: UPDATE POINTER
1685	006222	000752			BR	2\$: BRANCH BACK TO CHECK NEXT REGISTER
1686								
1687								
1688	006224	005067	173532	MANUAL:	CLR	FSTADR		: INIT FIRST ADDRESS
1689	006230	105767	173522		TSTB	SELFLG		: CHECK FOR SELECT PARAMETER SETUP
1690	006234	001002			BNE	MANUL1		: IF FLAG SET GET USERS PARAMETERS
1691	006236	000167	000402		JMP	MANUL2		: ELSE USE DEFAULT DATA
1692	006242			MANUL1:				
1693	006242	004567	013326		JSR	R5,	\$PRINT	: GO PRINT OUT THE FOLLOWING MESSAGE.
1694	006246	024275			.WORD	FADMES		: ADDRESS OF MESSAGE TO BE TYPED
1695								: 'FIRST ADDRESS:'
1696								: FIRST ADDRESS 8K BOUNDARY
1697								
1698								
1699	006250	106746						: * THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE \$RDOCT ROUTINE
1700	006252	105066	000001					: * WIHTOUT USING A 'TRAP' INSTRUCTION AS CALLED FOR BY **SYSMAC**.
1701					MFPS	-(SP)		: PUT THE PROCESSOR STATUS ON THE STACK
1702	006256	004767	013136		CLRB	1(SP)		: HIGH BYTE CLEARED TO INSURE KERNEL MODE
1703	006262	032716	037777					: ON PSW RETURN.
1704	006266	001365			JSR	PC,	\$RDOCT	: GO TO THE SUBROUTINE
1705	006270	016703	013276		BIT	#37777,(SP)		: ON 8K BOUNDARY?
1706	006274	011604			BNE	MANUL1		: IF NO,REASK
1707	006276	006267	013270		MOV	\$HI0CT,R3		: SAVE ORIGINAL HIGH BITS
1708	006302	006016			MOV	(SP),R4		: AND LOW BITS
1709	006304	006267	013262		ASR	\$HI0CT		: DIVIDE HIGH ADDRESS INTO
1710	006310	006016			ROR	(SP)		: NUMBER OF 128K BANKS AND
1711	006312	006367	013254		ASR	\$HI0CT		: NUMBER OF 8K BANK WITHIN
1712	006316	012700	001714		ROR	(SP)		: 128K BANK
1713	006322	066700	013244		ASL	\$HI0CT		: MAKE NUMBER OF 128K BANKS INDEX
1714	006326	012701	000012		MOV	#BITPT,R0		: GET START OF TEST TABLE
1715	006332	006016			ADD	\$HI0CT,R0		: ADD INDEX
1716	006334	077102		1\$:	MOV	#12,R1		: SET UP TO ALIGN 8K BANK COUNT
1717	006336	052710	000001		ROR	(SP)		: ALIGN COUNT
1718	006342	005716			SOB	R1,1\$: LOOP UNTIL DONE
1719	006344	001403			BIS	#BIT0,(R0)		: INIT 8K BANK POINTER IN 128K WORD
1720	006346	006310		2\$:	TST	(SP)		: SHIFT BIT UNTIL COUNT = 0
1721	006350	005316			BEQ	3\$: WHEN ZERO DONE
1722	006352	000773			ASL	(R0)		: SHIFT BIT
1723	006354	012702	001514		DEC	(SP)		: SUBTRACT FROM COUNT
1724	006360	066702	013206	3\$:	BR	2\$: LOOP BACK
					MOV	#MEMMAP,R2		: GET FIRST ADDRESS OF MEMORY TABLE
					ADD	\$HI0CT,R2		: ADD INDEX

Address	OpCode	Operand 1	Operand 2	Label	Instruction	Comments
1725	006364	031012			BIT (R0),(R2)	:IS BIT JUST LOCATED,SET IN MEM. TABLE
1726	006366	001725			BEQ MANUL1	:IF NO BANK EXISTS...REASK
1727	006370			4\$:		
1728	006370	004567	013200		JSR R5, \$PRINT	:GO PRINT OUT THE FOLLOWING MESSAGE.
1729	006374	024362			.WORD LADMES	:ADDRESS OF MESSAGE TO BE TYPED
1730						:LAST ADDRESS:"
1731						:* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE \$RDOCT ROUTINE
1732						:* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
1733	006376	106746			MFPS -(SP)	:PUT THE PROCESSOR STATUS ON THE STACK
1734	006400	105066	000001		CLRB 1(SP)	:HIGH BYTE CLEARED TO INSURE KERNEL MODE
1735						:ON PSW RETURN.
1736	006404	004767	013010		JSR PC, \$RDOCT	:GO TO THE SUBROUTINE
1737	006410	005716			TST (SP)	:CHECK IF LOW BITS ZERO
1738	006412	001010			BNE 5\$:IF NOT,NO DEFAULT
1739	006414	005767	013152		TST \$HIOCT	:CHECK IF HIGH BITS ZERO
1740	006420	001005			BNE 5\$:IF NOT,NO DEFAULT
1741	006422	016716	172526		MOV \$TMP2,(SP)	:IF BOTH ZERO,FILL IN DEFAULT
1742	006426	016767	172524	013136	MOV \$TMP3,\$HIOCT	:FOR LOW AND HIGH BITS
1743	006434	020467	013132	5\$:	CMP R4,\$HIOCT	:CHECK FOR LAST ADDR. BELOW FIRST
1744	006440	101353			BHI 4\$:IF YES....REASK
1745	006442	103402			BLO 6\$:IF LAST HIGHER GO ON
1746	006444	020316			CMP R3,(SP)	:IF EQUAL CHECK LOW BITS
1747	006446	101350			BHI 4\$:IF LOW BITS LOWER...REASK
1748	006450	012700	000020	6\$:	MOV #20,R0	:TABLE COUNTER
1749	006454	012701	001714		MOV #BITPT,R1	
1750	006460	012702	001614		MOV #SAVTST,R2	
1751	006464	052122		7\$:	BIS (R1)+,(R2)+	:STORE BITPT IN SAVTST MAP
1752	006466	077002			SOB R0,7\$:DO 16. TIMES
1753	006470	020367	013076		CMP R3,\$HIOCT	:COMPARE HIGH BITS OF FIRST TO LAST
1754	006474	103403			BLO 8\$:IF LOWER SEE IF NEXT BANK EXISTS
1755	006476	101027			BHI 11\$:IF HIGHER MUST BE LAST BANK
1756	006500	020416			CMP R4,(SP)	:IF EQUAL CHECK LOW BITS
1757	006502	101025			BHI 11\$:IF LOW BITS FIRST HIGHER LAST BANK
1758	006504	062704	040000	8\$:	ADD #40000,R4	:UPDATE TO NEXT BANK
1759						
1760	006510	005503			ADC R3	
1761	006512	012700	000017		MOV #17,R0	:SET UP TO UPDATE POINTER
1762	006516	012701	001714		MOV #BITPT,R1	:GET START ADDRESS OF TABLE
1763	006522	006321			ASL (R1)+	:ROTATE POINTER WITHIN
1764	006524	006121		9\$:	ROL (R1)+	:TABLE
1765	006526	077002			SOB R0,9\$:UNTIL ALL LOCATIONS DONE
1766	006530	103423			BCS 13\$:BANK DOESN'T MAP
1767	006532	012700	000020		MOV #20,R0	:SET UP TO SEE IF ONE EXISTS
1768	006536	012701	001714		MOV #BITPT,R1	:GET START ADDRESS OF POINTER TABLE
1769	006542	012702	001514		MOV #MEMMAP,R2	:GET START ADDRESS OF MEMORY TABLE
1770	006546	032122		10\$:	BIT (R1)+,(R2)+	:TEST IF BANK EXISTS
1771	006550	001337			BNE 6\$:IF MATCH UPDATE
1772	006552	077003			SOB R0,10\$:DO ALL OF TABLE
1773	006554	000753			BR 8\$	
1774	006556	012700	000020	11\$:	MOV #20,R0	:MAKE SURE LAST BANK
1775	006562	012701	001714		MOV #BITPT,R1	:WAS MAPPED BY MEMORY
1776	006566	012702	001514		MOV #MEMMAP,R2	:SIZING ROUTINE
1777	006572	032122		12\$:	BIT (R1)+,(R2)+	:
1778	006574	001007			BNE 14\$:
1779	006576	077003			SOB R0,12\$:
1780	006600	012706	001100	13\$:	MOV #STACK,SP	:RESET STACK


```

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

```


1867
1868
1869
1870
1871
1872
1873
1874
1875
1876 007100
1877 007100 004567 010250
1878
1879 007104 004467 003670
1880 007110 010200
1881 007112 004767 005002
1882 007116 006305
1883 007120 110022
1884 007122 005200
1885 007124 077503
1886 007126 004767 004060
1887
1888
1889
1890 007132 004467 003740
1891 007136 010200
1892 007140 005300
1893 007142 004767 004752
1894 007146 006305
1895 007150 114201
1896 007152 120001
1897 007154 001405
1898 007156 004767 007416
1899 007162 004767 010500
1900 007166 000003
1901 007170
1902 007170 005300
1903 007172 077512
1904 007174 004767 004356
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915 007200
1916 007200 004567 010150
1917
1918 007204 004467 003666
1919 007210 010200
1920 007212 162700 000002
1921 007216 004767 004676
1922 007222 005100

```
*****  
*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.  
*****  
TST2:  
JSR R5, $SCOPE ;GO TO SCOPE ROUTINE.  
;* UPWARDS BYTE ADDRESSING.  
JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.  
1$: MOV R2, R0 ;SET UP TO CALCULATE PHYSICAL ADDRESS  
JSR PC, PHYADR ;GET PHYSICAL ADDRESS INTO R0  
ASL R5 ;MAKE TEST COUNTER BYTE VALUE  
2$: MOVB R0, (R2)+ ;WRITE VALUE OF ADDRESS INTO ADDRESS  
INC R0 ;ADD ONE TO PHYSICAL ADDRESS  
SOB R5, 2$ ;BRANCH IF MORE IN CURRENT BLOCK.  
JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 1$.  
  
;* CHECK THAT VALUE OF MEMORY ADDRESS WAS WRITTEN CORRECTLY  
;* DOWNWARDS BYTE ADDRESSING.  
JSR R4, INITDN ;INITIALIZE THE MEMORY ADDRESS POINTERS.  
3$: MOV R2, R0 ;SET UP TO CALCULATE PHYSICAL ADDRESS  
DEC R0 ;GET LAST BYTE ADDRESS OF BLOCK  
JSR PC, PHYADR ;GET PHYSICAL ADDRESS INTO R0  
ASL R5 ;MAKE TEST COUNTER BYTE VALUE  
4$: MOVB -(R2), R1 ;GET THE DATA FROM MEMORY  
CMPB R0, R1 ;CHECK THE DATA...LO BYTE ONLY VALID.  
BEQ 65$ ;BRANCH OVER ERROR CALL IF GOOD DATA.  
64$: JSR PC, SPRNTO ;SET UP VALUES FOR ERROR PRINTING.  
JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)  
.WORD 3 ;ERROR TYPE CODE.  
65$: DEC R0 ;DEC DATA BY 1  
SOB R5, 4$ ;BRANCH IF MORE IN CURRENT BLOCK.  
JSR PC, MMDOWN ;FIND NEXT BLOCK AND LOOP TO 3$.  
  
*****  
*TEST 3 WRITE 1'S COMPLEMENT VALUE OF ADDRESS INTO ADDRESS.  
* R0 = DATA WRITTEN INTO MEMORY (SHOULD BE)  
* R1 = DATA READ FROM MEMORY (WAS)  
* R2 = VIRTUAL ADDRESS  
* R3 = NOT USED  
* R4 = NOT USED  
* R5 = BLOCK BOUNDARY BIT MASK.  
*****  
TST3:  
JSR R5, $SCOPE ;GO TO SCOPE ROUTINE.  
;* DOWNWARDS WORD ADDRESSING.  
JSR R4, INITDN ;INITIALIZE THE MEMORY ADDRESS POINTERS.  
1$: MOV R2, R0 ;SET UP TO CALCULATE PHYSICAL ADDRESS  
SUB #2, R0 ;GET LAST ADDRESS OF BLOCK  
JSR PC, PHYADR ;GET PHYSICAL ADDRESS INTO R0  
COM R0 ;COMPLEMENT THE ADR
```

```

1923 007224 010042
1924 007226 062700 000002
1925 007232 077504
1926 007234 004767 004316
1927
1928
1929
1930 007240 004467 003534
1931 007244 010200
1932 007246 004767 004646
1933 007252 005100
1934 007254
1935 007254 012201
1936 007256 020001
1937 007260 001405
1938 007262 004767 007336
1939 007266 004767 010374
1940 007272 000002
1941 007274
1942 007274 162700 000002
1943 007300 077513
1944 007302 004767 003704
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955 007306
1956 007306 004567 010042
1957
1958 007312 004467 003462
1959 007316 004767 004674
1960 007322 006305
1961 007324 110022
1962 007326 077502
1963 007330 004767 003656
1964
1965
1966
1967 007334 004467 003440
1968 007340 004767 004652
1969 007344 006305
1970 007346 112201
1971 007350 120001
1972 007352 001405
1973 007354 004767 007226
1974 007360 004767 010302
1975 007364 000003
1976 007366
1977 007366 077511
1978 007370 004767 003616
    
```

```

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.
    JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 1$.

;* CHECK COMPLEMENT DATA WRITTEN DOWN
;* UPWARDS WORD ADDRESSING.
3$: JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
    MOV R2, R0 ;SET UP TO CALCULATE PHYSICAL ADDRESS
    JSR PC, PHYADR ;GET PHYSICAL ADDRESS INTO R0
    COM R0 ;COMPLEMENT IT

4$: MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
    CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
    BEQ 65$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
64$: JSR PC, SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
    JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
    .WORD 2 ;ERROR TYPE CODE.

65$: SUB #2, R0 ;COUNT DOWN WITH ADDRESS
    SOB R5, 4$ ;BRANCH IF MORE IN CURRENT BLOCK
    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.
*****
TST4: JSR R5, $SCOPE ;GO TO SCOPE ROUTINE.
;* UPWARDS BYTE ADDRESSING.
1$: JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
    JSR PC, BANKNO ;GET THE BANK NUMBER INTO R0
    ASL R5 ;MAKE TEST COUNTER BYTE VALUE
2$: MOVB R0, (R2)+ ;WRITE BANK # INTO ALL ADDRESSES
    SOB R5, 2$ ;BRANCH IF MORE IN CURRENT BLOCK.
    JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 1$.

;* CHECK THAT DATA WRITTEN ABOVE CAN BE READ
;* UPWARDS BYTE ADDRESSING.
3$: JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
    JSR PC, BANKNO ;GET THE BANK NUMBER INTO R0
    ASL R5 ;MAKE TEST COUNTER BYTE VALUE
4$: MOVB (R2)+, R1 ;READ THE DATA OUT OF MEMORY
    CMPB R0, R1 ;CHECK DATA...LOW BYTE ONLY VALID
    BEQ 65$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
64$: JSR PC, SPRNT1 ;SET UP VALUES FOR ERROR PRINTING.
    JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
    .WORD 3 ;ERROR TYPE CODE.

65$: SOB R5, 4$ ;BRANCH IF MORE IN CURRENT BLOCK
    JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 3$.
    
```


1979
 1980
 1981
 1982
 1983
 1984
 1985
 1986
 1987
 1988
 1989 007374
 1990 007374 004567 007754
 1991
 1992 007400 004467 003472
 1993 007404 004767 004606
 1994 007410 006305
 1995 007412 005100
 1996 007414 110042
 1997 007416 077502
 1998 007420 004767 004132
 1999
 2000
 2001
 2002 007424 004467 003446
 2003 007430 004767 004562
 2004 007434 006305
 2005 007436 005100
 2006 007440 114201
 2007 007442 120001
 2008 007444 001405
 2009 007446 004767 007126
 2010 007452 004767 010210
 2011 007456 000003
 2012 007460
 2013 007460 077511
 2014 007462 004767 004070
 2015
 2016
 2017
 2018
 2019
 2020
 2021
 2022
 2023 007466 032767 000001 171040
 2024 007474 001014
 2025 007476 012700 000020
 2026 007502 016701 172302
 2027 007506 005021
 2028 007510 077002
 2029 007512 016700 172076
 2030 007516 042700 177776
 2031 007522 010067 172026
 2032 007526
 2033
 2034

```

*****
: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.
*****
TST5:
      JSR      R5,      $SCOPE ;GO TO SCOPE ROUTINE.
:* DOWNWARDS BYTE ADDRESSING.
      JSR      R4,      INITDN ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$:   JSR      PC,      BANKNO ;GET THE BANK NUMBER INTO R0
      ASL      R5
      COM      R0 ;MAKE TEST COUNTER BYTE VALUE
      ;1'S COMPLEMENT OF BANK #
2$:   MOVB    R0,      -(R2) ;PUT 1'S COM OF BANK # INTO MEMORY
      SOB     R5,      2$ ;BRANCH IF MORE IN CURRENT BLOCK.
      JSR      PC,      MMDOWN ;FIND NEXT BLOCK AND LOOP TO 1$.

:* CHECK THAT DATA WRITTEN CAN BE READ.
:* DOWNWARDS BYTE ADDRESSING.
      JSR      R4,      INITDN ;INITIALIZE THE MEMORY ADDRESS POINTERS.
3$:   JSR      PC,      BANKNO ;GET THE BANK # INTO R0
      ASL      R5
      COM      R0 ;MAKE TEST COUNTER BYTE VALUE
      ;SET 1'S COMPLEMENT OF BANK #
4$:   MOVB    -(R2),   R1 ;READ DATA OUT OF MEMORY
      CMPB    R0,      R1 ;CHECK DATA...LOW BYTE ONLY VALID
      BEQ     65$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
64$:  JSR      PC,      SPRNTO ;SET UP VALUES FOR ERROR PRINTING.
      JSR      PC,      $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
      .WORD   3 ;ERROR TYPE CODE.
65$:  SOB     R5,      4$ ;BRANCH IF MORE IN CURRENT BLOCK.
      JSR      PC,      MMDOWN ;FIND NEXT BLOCK AND LOOP TO 3$.

:* IF PROGRAM HAS RELOCATED TO UPPER BOUNDARY,THE ADDRESSING
:* TESTS WILL BE EXECUTED FOR ALL BANKS AND WORST CASE NOISE TESTING
:* WILL BE LIMITED TO BANKS 0,1.
:* ALL OTHER BANKS WILL BE EXERCISED FOR WORST CASE NOISE TESTING
:* WHEN THE PROGRAM OCCUPIES BANKS 0,1.
SECT2: BIT #BIT0,PRGMAP ;IS PROGRAM RELOCATED
      BNE     2$ ;IF NO,CONTINUE TESTING ALL BANKS
      MOV     #20,R0 ;CLEAR ALL LOCATIONS OF TEST MAP
      MOV     .TSTMAP,R1 ;TEST MAP TABLE
1$:   CLR     (R1)+
      SOB    R0,1$ ;LOOP UNTIL DONE
      MOV     SAVTST,R0 ;GET LOWER 128K MAP
      BIC     #177776,R0 ;DO ONLY FIRST BANK
      MOV     R0,TSTMAP ;FOR DATA TESTS
2$:   ;CONTINUE TESTING
    
```

2035
 2036
 2037
 2038
 2039
 2040
 2041
 2042
 2043
 2044
 2045
 2046
 2047
 2048
 2049
 2050 007526
 2051 007526 004567 007622
 2052 007532 016700 172226
 2053 007536 004467 003236
 2054 007542 010022
 2055 007544 077502
 2056 007546 004767 003440
 2057
 2058
 2059
 2060
 2061
 2062 007552
 2063 007552 004567 007576
 2064 007556 016700 172202
 2065 007562 004467 003212
 2066 007566
 2067 007566 012201
 2068 007570 020001
 2069 007572 001405
 2070 007574 004767 007024
 2071 007600 004767 010062
 2072 007604 000004
 2073 007606
 2074 007606 077511
 2075 007610 004767 003376
 2076
 2077
 2078
 2079 007614 032777 000400 171306
 2080 007622 001416
 2081 007624 017746 171300
 2082 007630 042716 177740
 2083 007634 022726 000006
 2084 007640 001007
 2085 007642 162767 000001 171222
 2086 007650 162767 000024 171220
 2087 007656 000725
 2088
 2089
 2090

```

.SBTTL SECTION 2:      WORST CASE NOISE TESTS
*****
* THESE TESTS WRITE MEMORY WORST CASE NOISE TEST PATTERNS THROUGHOUT
* MEMORY AND CHECK THAT THEY CAN BE WRITTEN AND READ.
*****
*TEST 6      WRITE A CONSTANT INTO MEMORY.
* THE CONSTANT IS USER SELECTABLE (DEFAULT = 0).
* R0 = DATA WRITTEN INTO MEMORY (SHOULD BE)
* R1 = DATA READ FROM MEMORY (WAS)
* R2 = VIRTUAL ADDRESS
* R3 = NOT USED
* R4 = NOT USED
* R5 = BLOCK BOUNDARY BIT MASK.
*****
TST6:
  JSR   R5,    $SCOPE ;GO TO SCOPE ROUTINE.
TST6A: MOV   .CONST, R0 ;GET USER CONSTANT
  JSR   R4,    INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$:   MOV   (R2)+, R1 ;WRITE CONSTANT INTO MEMORY.
  SOB   R5,    1$ ;BRANCH IF MORE IN CURRENT BLOCK
  JSR   PC,    MMUP  ;FIND NEXT BLOCK AND LOOP TO 1$.
*****
*TEST 7      READ MEMORY AND COMPARE TO CONSTANT.
* IMPORTANT: THIS TEST SHOULD NOT BE RUN WITHOUT FIRST RUNNING TEST $TN.
*****
TST7:
  JSR   R5,    $SCOPE ;GO TO SCOPE ROUTINE.
  MOV   .CONST, R0 ;GET USER CONSTANT
  JSR   R4,    INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$:   MOV   (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
  CMP   R0,    R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
  BEQ   65$,   ;BRANCH OVER ERROR CALL IF GOOD DATA.
64$:  JSR   PC,    SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
  JSR   PC,    $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
  .WORD 4 ;ERROR TYPE CODE.
65$:  SOB   R5,    1$ ;BRANCH IF MORE IN CURRENT BLOCK
  JSR   PC,    MMUP  ;FIND NEXT BLOCK AND LOOP TO 1$.
* SPECIAL CHECK TO SEE IF TEST 6 IS SELECTED THRU THE SWR.
* ALLOWS THE OPERATOR TO SWITCH BACK AND FORTH BETWEEN TESTS 6 AND 7
* BY SIMPLY "TOGGLING" SW00 WHEN SW01, SW02, AND SW08 ARE SET.
  BIT   #SW08, @SWR ;CHECK THAT LOOP ON TEST BIT SET
  BEQ   TST10 ;BRANCH IF NOT LOOP ON TEST
  MOV   @SWR, -(SP) ;GET SWITCH REGISTER DATA.
  BIC   #177740, (SP) ;CLEAR NON-TEST-NUMBER SWITCHES.
  CMP   #6,    (SP)+ ;CHECK IF TEST 6 IN SWITCHES.
  BNE   TST10 ;BRANCH IF NOT TEST 6
  SUB   #1,    $TSTNM ;RESET TEST NUM
  SUB   #TST7-TST6,$LPADR ;RESET LOOP ADR
  BR    TST6A ;GO TO TEST 6
*****
*TEST 10     WORSE CASE NOISE (PARITY) WORD TESTING

```


;* CHECK MEMORY WITH A SERIES OF PATTERNS

2091
2092
2093 007660
2094 007660 004567 007470
2095 007664 016704 172136
2096 007670 004767 005766
2097 007674 012400
2098 007676 001417
2099 007700 004467 003074
2100 007704 010012
2101 007706 012201
2102 007710 020001
2103 007712 001405
2104 007714 004767 006704
2105 007720 004767 007742
2106 007724 000004
2107 007726
2108 007726 077512
2109 007730 004767 003256
2110 007734 000755

TST10:
1\$: JSR R5, \$SCOPE ;GO TO SCOPE ROUTINE.
MOV .MPPAT, R4 ;INITIALIZE PATTERN TABLE POINTER
JSR PC, CKPMER ;CHECK FOR NON-TRAP PARITY MEMORY ERRORS.
MOV (R4)+, R0 ;GET THE DATA PATTERN.
BEQ TST11 ;BR IF END OF TABLE.
2\$: JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
MOV R0, (R2) ;PUT DATA PATTERN INTO MEMORY.
MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
BEQ 65\$;BRANCH OVER ERROR CALL IF GOOD DATA.
64\$: JSR PC, SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
JSR PC, \$ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
.WORD 4 ;ERROR TYPE CODE.
65\$: SOB R5, 2\$;BRANCH IF MORE IN CURRENT BLOCK
JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 2\$.
BR 1\$;BR BACK TO DO NEXT PATTERN

2111
2112
2113
2114 007736
2115 007736 004567 007412
2116 007742 012700 177777
2117 007746 004767 004332
2118 007752 004467 003022
2119 007756 000241
2120 007760 004767 004336
2121 007764 016201 177776
2122 007770 103402
2123 007772 020001
2124 007774 001405
2125 007776 004767 006622
2126 010002 004767 007660
2127 010006 000005
2128 010010
2129 010010 077516
2130 010012 004767 003174
2131
2132
2133
2134
2135 010016
2136 010016 004567 007332
2137 010022 005000
2138 010024 004767 004254
2139 010030 004467 002744
2140 010034 000261
2141 010036 004767 004260
2142 010042 016201 177776
2143 010046 103002
2144 010050 020001
2145 010052 001405
2146 010054 004767 006544
2147 010060 004767 007602
2148 010064 000005
2149 010066
2150 010066 077516
2151 010070 004767 003116
2152

```
*****  
:TEST 11 ROTATE A '0' BIT THROUGH A FIELD OF ONES.  
*****  
TST11:  
JSR R5, $SCOPE ;GO TO SCOPE ROUTINE.  
MOV #-1, R0 ;SET CHECK WORD  
JSR PC, SETCON ;PUT THE CONTENTS OF R0 IN ALL MEMORY.  
JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.  
1$: CLC ;CLEAR CARRY BIT IN PSW  
JSR PC, ROTATE  
MOV -2(R2), R1 ;GET RESULT  
BCS 63$ ;BRANCH IF 'C' BIT WAS SET  
CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.  
BEQ 64$ ;BRANCH OVER ERROR CALL IF GOOD DATA.  
63$: JSR PC, SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.  
JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)  
.WORD 5 ;ERROR TYPE CODE.  
64$: SOB R5, 1$ ;BRANCH IF MORE IN CURRENT BLOCK  
JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 1$.
```

```
*****  
:TEST 12 ROTATE A '1' BIT THROUGH A FIELD OF ZEROS  
*****  
TST12:  
JSR R5, $SCOPE ;GO TO SCOPE ROUTINE.  
CLR R0 ;SET CHECK WORD  
JSR PC, SETCON ;PUT THE CONTENTS OF R0 IN ALL MEMORY  
JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.  
1$: SEC ;SET 'C' BIT IN PSW  
JSR PC, ROTATE ;GO ROTATE '1' BIT  
MOV -2(R2), R1 ;GET RESULT  
BCC 63$ ;BRANCH IF 'C' IS CLEAR  
CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.  
BEQ 64$ ;BRANCH OVER ERROR CALL IF GOOD DATA.  
63$: JSR PC, SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.  
JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)  
.WORD 5 ;ERROR TYPE CODE.  
64$: SOB R5, 1$ ;BRANCH IF MORE IN CURRENT BLOCK  
JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 1$.
```


2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163 010074
2164 010074 004567 007254
2165 010100 005767 173264
2166 010104 001404
2167 010106 032777 000100 171014
2168 010114 001402
2169 010116 000167 000742
2170 010122 005000
2171 010124 004767 004154
2172 010130 004467 002644
2173
2174
2175 010134 006305
2176 010136 012700 000020
2177 010142 016701 171646
2178 010146 016703 171640
2179 010152 032123
2180 010154 001004
2181 010156 077003
2182 010160 060502
2183
2184 010162 000167 000666
2185 010166 004767 005434
2186 010172 004767 005464
2187 010176 020227 000114
2188 010202 001004
2189 010204 062702 000004
2190 010210 162705 000004
2191
2192 010214 111201
2193 010216 001405
2194 010220 004767 006312
2195 010224 004767 007436
2196 010230 000011
2197 010232
2198 010232 105067 171522
2199 010236 112700 000252
2200 010242 110012
2201 010244 016703 171552
2202 010250 056773 171512 000000
2203 010256 052733 000001
2204 010262 005713
2205 010264 001371
2206
2207 010266 110012
2208 010270 016703 171526

```

*****
*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.
*****
TST13:
WWPB0: JSR      R5,      $SCOPE ;GO TO SCOPE ROUTINE.
        TST      MPRX    ;CHECK FOR ANY PARITY MEMORY.
        BEQ      1$,    ;BR IF NO PARITY MEMORY.
        BIT      #SW06, @SWR  ;CHECK FORINHIBIT PARITY SWITCH.
        BEQ      2$,    ;BR IF NOT SET.
1$:     JMP      TST14   ;SKIP THIS TEST IF NO PARITY MEMORY PRESENT.
2$:     CLR      R0      ;ZERO TO BE PUT IN ALL MEMORY.
        JSR      PC,    SETCON ;ROUTINE TO LOAD ALL MEMORY.
        JSR      R4,    INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.

;RETURN FOR MMUP
;MAKE TEST COUNTER BYTE VALUE
;TABLE COUNT
;BANK POINTER TABLE
;PARITY TABLE
WWPBYT: ASL      R5
        MOV      #20,   R0
        MOV      .BITPT, R1 ;CHECK IF CURRENT BANK HAS PARITY MEMORY
1$:     MOV      .PMEMAP,R3 ;BRANCH IF PARITY MEMORY
        BIT      (R1)+,(R3)+ ;CHECK TILL END OF TABLE
        BNE     2$,    ;IF NONE FOUND POINT R2 TO
        SOB     R0,    1$ ;FIRST ADDR OF NEXT BANK.
        ADD     R5,    R2 ;FIND NEXT BLOCK
2$:     JMP      WWPB6
        JSR      PC,    SETAE  ;SET ACTION ENABLE (EVEN IF BANK0.)
        JSR      PC,    CKPMER ;CHECK FOR ANY NON TRAP PARITY ERRORS.
WWPB1:  CMP      R2,    #114   ;CHECK IF POINTING TO PARITY ERROR VECTOR.
        BNE     3$,    ;BR IF NOT AT VECTOR.
        ADD     #4,    R2     ;SKIP PARITY VECTOR.
        SUB     #4,    R5     ;SKIP PARITY VECTOR
        ;SUBTRACT 4 FROM TEST COUNTER
3$:     MOVB    (R2),   R1     ;CHECK IF BYTE STILL CLEARED.
        BEQ     65$,    ;BRANCH OVER ERROR CALL IF GOOD DATA.
64$:    JSR      PC,    SPRNT  ;SET UP VALUES FOR ERROR PRINTING.
        JSR      PC,    $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
        .WORD   11          ;ERROR TYPE CODE.
65$:    CLRB    OEFLG   ;CLEAR ODD/EVEN FLAG.
        MOVB   #252,   R0     ;SET UP DATA...EVEN, SETS PARITY BIT.
WWPB2:  MOVB    R0,    (R2)   ;MOV DATA INTO TEST LOCATION.
        MOV    .MPRX,  R3     ;GET PARITY REGISTER TABLE POINTER.
10$:    BIS     WWP,@(R3) ;SET WRITE WRONG PARITY.
        BIS    #AE,@(R3)+
        TST    (R3)
        BNE    10$
; * SET WRONG PARITY IN LOCATION UNDER TEST.
        MOVB   R0,    (R2)   ;WRITE SAME DATA (EXCEPT PARITY) VIA DATOB.
        MOV    .MPRX,  R3     ;GET PARITY REG TABLE POINTER.

```

```

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



```

2265 010514 001402          BEQ      25$          ;BR IF R2 IS ADDRESSING PAR2
2266 010516 062700 000200    ADD      #200,R0      ;SET UP EXPECTED WITH PAR3
2267 010522 052700 100001    25$:    BIS      #BIT15+BIT0,R0 ;SET ERROR AND AE BIT IN CHECK WORD.
2268 010526 016367 000042    170752  MOV      42(R3),RESRVD ;GET APPROPRIATE MASK.
2269 010534 046700 170746    BIC      RESRVD,R0    ;CLEAR PARITY REG BITS RESERVED FOR FUTURE.
2270 010540 046701 170742    BIC      RESRVD,R1    ;CLEAR PARITY REG BITS RESERVED FOR FUTURE.
2271                                ;NOTE: THE ABOVE INSTRUCTION (2 WORDS) CAN BE NOP'ED FOR UNMIXED MEMORY TYPES.
2272 010544 020001          CMP      R0,R1        ;COMPARE THE CHECK WORD WITH THE DATA READ.
2273 010546 001405          BEQ      67$          ;BRANCH OVER ERROR CALL IF GOOD DATA.
2274 010550 004767 006020    66$:    JSR      PC,SPRNTP    ;SET UP VALUES FOR ERROR PRINTING.
2275 010554 004767 007106    JSR      PC,$ERROR    ;*** ERROR *** (GO TYPE A MESSAGE)
2276 010560 000015          .WORD   15           ;ERROR TYPE CODE.
2277 010562
2278 010562 005767 167750    67$:    TST      MMAVA        ;CHECK FOR 22 BIT ADDRESSING
2279 010566 100034          BPL      100$         ;BR IF 22 BIT ADDR NOT ENABLED
2280 010570 022737 010000 172344  CMP      #10000,@#KIPAR2 ;CHECK IF ABOVE 128KW
2281 010576 101030          BHI      100$         ;BR IF KIPAR2 LESS THAN 128KW
2282 010600 010046          MOV      R0,-(SP)     ;:PUSH R0 ON STACK
2283 010602 052773 040000 000000  BIS      #BIT14,@(R3) ;SET BIT14 TO ENABLE BITS 18-21
2284                                ;INTO BITS 8-5 OF PARITY CSR
2285 010610 013700 172344    MOV      @#KIPAR2,R0  ;SETUP EXPECTED FOR PARITY CSR
2286 010614 042700 007777    BIC      #7777,R0     ;BITS 18-21 STORED IN
2287 010620 000300          SWAB     R0           ;BITS 5-8.
2288 010622 006300          ASL      R0           ;A18-21 IN BITS 5-8
2289 010624 052700 140001    BIS      #BIT15!BIT14!AE,R0 ;SAVE EXPECTED PARITY CSR
2290 010630 017301 000000    MOV      @(R3),R1     ;GET CONTENTS OF PARITY CSR
2291 010634 042701 030032    BIC      #30032,R1    ;CLEAR MASKED BITS
2292 010640 020001          CMP      R0,R1        ;COMPARE THE CHECK WORD WITH THE DATA READ.
2293 010642 001405          BEQ      69$          ;BRANCH OVER ERROR CALL IF GOOD DATA.
2294 010644 004767 005724    68$:    JSR      PC,SPRNTP    ;SET UP VALUES FOR ERROR PRINTING.
2295 010650 004767 007012    JSR      PC,$ERROR    ;*** ERROR *** (GO TYPE A MESSAGE)
2296 010654 000015          .WORD   15           ;ERROR TYPE CODE.
2297 010656
2298 010656 012600          MOV      (SP)+,R0     ;:POP STACK INTO R0
2299 010660 005073 000000    100$:   CLR      @(R3)        ;CLEAR REG INCLUDING ACTION ENABLE.
2300 010664 010346          MOV      R3,-(SP)    ;:PUSH R3 ON STACK
2301 010666 062703 000044    26$:    ADD      #44,R3       ;UPDATE POINTER TO NEXT PARITY REG + MAP.
2302 010672 020367 171124    CMP      R3,.MPRX    ;CHECK FOR END OF TABLE.
2303 010676 101014          BHI      WWPB3        ;BR IF END OF TABLE REACHED.
2304 010700 032713 000001    BIT      #BIT0,(R3)  ;CHECK IF NEXT REG EXISTS.
2305 010704 001370          BNE      26$         ;BR IF THIS PARITY REG DOESN'T EXIST.
2306 010706 017301 000000    MOV      @(R3),R1    ;SAVE AND CHECK FOR ERROR FLAG.
2307 010712 100365          BPL      26$         ;BR IF NO ERROR FLAG.
2308 010714 004767 005654    70$:    JSR      PC,SPRNTP    ;SET UP VALUES FOR ERROR PRINTING.
2309 010720 004767 006742    JSR      PC,$ERROR    ;*** ERROR *** (GO TYPE A MESSAGE)
2310 010724 000016          .WORD   16           ;ERROR TYPE CODE.
2311 010726 000757          BR       26$         ;BR AFTER ERROR.
2312 010730 111204    WWPB3:  MOVB    (R2),R4      ;GET THE DATA FOR CHECKING.
2313                                ;* READING THE DATA VIA DAT1 TO CHECK IT SHOULD CAUSE PARITY ERROR, BUT
2314                                ;* ACTION ENABLE IS NOT SET IN CONTROLLING REG, SO NO TRAP SHOULD OCCUR.
2315 010732 111212    MOVB    (R2),R2      ;RESTORE RIGHT PARITY
2316                                ;NOTE: THE ABOVE INSTRUCTION CAN BE NOP'ED FOR PROCESSORS
2317                                ;WHICH DO ONLY DATOB TO DESTINATION OF MOVB INSTRUCTIONS.
2318 010734 012603          MOV      (SP)+,R3    ;:POP STACK INTO R3
2319 010736 017301 000000    MOV      @(R3),R1    ;:READ THE PARITY REGISTER TO CHECK IT AGAIN.
2320 010742 046701 170540    BIC      RESRVD,R1    ;CLEAR PARITY REG BITS RESERVED FOR FUTURE.

```

```

2321
2322 010746 042700 000001
2323 010752 020001
2324 010754 001405
2325 010756 004767 005612
2326 010762 004767 006700
2327 010766 000015
2328 010770
2329 010770 012773 000001 000000
2330 010776 010401
2331 011000 012600
2332 011002 120001
2333 011004 001405
2334 011006 004767 005566
2335 011012 004767 006650
2336 011016 000017
2337 011020
2338 011020 110012
2339 011022 105712
2340 011024 012700 000253
2341 011030 105167 170724
2342 011034 100002
2343 011036 000167 177200
2344 011042 005202
2345 011044 005305
2346 011046 001402
2347 011050 000167 177122
2348 011054 004767 002132
2349 011060 004767 004476

;NOTE: THE ABOVE INSTRUCTION (2 WORDS) CAN BE NOP'ED FOR UNMIXED MEMORY TYPES.
BIC #AE, R0 ;CLEAR THE ACTION ENABLE BIT IN TEST DATA.
CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
BEQ 65$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
64$: JSR PC, SPRNTP ;SET UP VALUES FOR ERROR PRINTING.
JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
.WORD 15 ;ERROR TYPE CODE.

65$: MOV #1, @(R3) ;CLEAR ALL BUT ACTION ENABLE.
MOV R4, R1 ;GET DATA READ FROM MEMORY FOR TESTING.
MOV (SP)+, R0 ;POP STACK INTO R0
CMPB R0, R1 ;CHECK THE DATA.
BEQ 67$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
66$: JSR PC, SPRNTO ;SET UP VALUES FOR ERROR PRINTING.
JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
.WORD 17 ;ERROR TYPE CODE.

67$: WWPB4: MOVB R0, (R2) ;RESTORE DATA.
TSTB (R2) ;DO A DATI TO BE SURE RIGHT PARITY.
MOV #253, R0 ;SET ODD PARITY DATA.
COMB DEFLG ;CHECK IF DONE BOTH ODD AND EVEN PARITY.
BPL 27$ ;BR IF DONE BOTH EVEN AND ODD.
27$: JMP WWPB2 ;LOOP BACK AND DO ODD(PARITY BIT CLR).
INC R2 ;MOVE POINTER TO NEXT MEMORY BYTE.
WWPB5: DEC R5 ;CHECK FOR END OF BLOCK
BEQ WWPB6 ;BR IF END OF BLOCK FOUND.
JMP WWPB1 ;LOOP BACK TO TEST NEXT BYTE.
WWPB6: JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO WWPBYT
JSR PC, MAMF ;GO RESET PARITY REGISTERS.
    
```


2350
 2351
 2352
 2353 011064
 2354 011064 004567 006264
 2355 011070 010703
 2356 011072 042703 007777
 2357 011076 004467 001676
 2358 011102 010246
 2359 011104 010346
 2360 011106 010567 170660
 2361 011112 012322
 2362 011114 032703 007777
 2363 011120 001002
 2364 011122 162703 010000
 2365 011126 077507
 2366 011130 012603
 2367 011132 012602
 2368 011134 016705 170632
 2369 011140 012300
 2370 011142 012201
 2371 011144 020001
 2372 011146 001405
 2373 011150 004767 005450
 2374 011154 004767 006506
 2375 011160 000020
 2376 011162
 2377 011162 032703 007777
 2378 011166 001002
 2379 011170 162703 010000
 2380 011174
 2381 011174 077517
 2382 011176 004767 002010
 2383
 2384
 2385
 2386
 2387
 2388
 2389
 2390
 2391
 2392
 2393
 2394
 2395
 2396
 2397
 2398
 2399
 2400
 2401
 2402
 2403
 2404
 2405

```

*****
*TEST 14      RANDOM DATA TESTING THRU PROGRAM CODE RELOCATION.
*****
TST14:
RANTST: JSR      R5,      $SCOPE ;GO TO SCOPE ROUTINE.
        MOV      PC,      R3      ;GET CURRENT PROGRAM COUNTER.
        BIC      #7777,   R3      ;POINT TO BEGINNING OF CURRENT 2K BLOCK.
        JSR      R4,      INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$:     MOV      R2,      -(SP)   ;SAVE MEMORY POINTER.
        MOV      R3,      -(SP)   ;SAVE 'DATA' POINTER.
        MOV      R5,      TEMP1  ;SAVE ADDRESS COUNTER
2$:     MOV      (R3)+,   (R2)+   ;MOV CODE INTO TEST MEMORY.
        BIT      #7777,   R3      ;CHECK FOR END OF 'DATA TABLE'
        BNE     3$        ;BRANCH IF MORE
        SUB     #10000,   R3      ;RESET POINTER TO START OF 'RANDOM DATA'
3$:     SOB     R5,      2$      ;IF NOT END OF BLOCK,BRANCH TO 2$
        MOV     (SP)+,   R3      ;RESET 'DATA' POINTER.
        MOV     (SP)+,   R2      ;RESET MEMORY POINTER.
        MOV     TEMP1,   R5      ;RESET ADDRESS COUNTER
4$:     MOV     (R3)+,   R0      ;GET S/B DATA.
        MOV     (R2)+,   R1      ;GET THE DATA FROM MEMORY UNDER TEST.
        CMP     R0,      R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
        BEQ    65$        ;BRANCH OVER ERROR CALL IF GOOD DATA.
64$:    JSR     PC,      SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
        JSR     PC,      $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
        .WORD   20          ;ERROR TYPE CODE.
65$:    BIT     #7777,   R3      ;CHECK FOR END OF 'DATA TABLE'
        BNE     5$        ;BR IF MORE.
        SUB     #10000,   R3      ;RESET POINTER TO TOP OF 'DATA TABLE'.
5$:     SOB     R5,      4$      ;BRANCH IF MORE IN CURRENT BLOCK
        JSR     PC,      MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
  
```

```

.SBTTL SECTION 3:      INSTRUCTION EXECUTION TESTS.
*****
*TEST 15      EXECUTE DATI, DATO THRU MEMORY.
* EXECUTES THE INSTRUCTION 'MOV R4,(R2)' THROUGHOUT MEMORY.
* AN 'RTS R5' (CODE 205) IS PLACED AFTER THE 'MOV' INSTRUCTION TO RETURN
* CONTROL TO THE MAIN PROGRAM FOR INSTRUCTION EXECUTION CHECKOUT.
* THIS IS AN EXAMPLE OF WHAT THIS TEST DOES IN RELATION TO MEMORY:
*
*          MEMORY LOCATION      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.
  
```

J 5

2406
 2407
 2408
 2409
 2410 011202
 2411 011202 004567 006146
 2412 011206 012703 010412
 2413 011212 012704 000205
 2414 011216 010400
 2415 011220 004467 001554
 2416 011224 005305
 2417 011226 010322
 2418 011230 010412
 2419 011232 004542
 2420 011234 012201
 2421 011236 020001
 2422 011240 001405
 2423 011242 004767 005352
 2424 011246 004767 006414
 2425 011252 000021
 2426 011254
 2427 011254 010322
 2428 011256 077514
 2429 011260 004767 001726

```

;* R3 = INSTRUCTION UNDER TEST (IUT).
;* R4 = RTS R5 (CODE 205).
;* R5 = BLOCK BOUNDARY BIT MASK.
:*****
TST15:
      JSR   R5,    $SCOPE ;GO TO SCOPE ROUTINE.
DIDO: MOV   #010412,R3 ;GET 'MOV R4,(R2)' INSTRUCTION (IUT).
      MOV   #205,  R4 ;GET 'RTS R5'
      MOV   R4,    R0 ;SET UP S/B DATA AFTER EXECUTION.
      JSR   R4,    INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$:   DEC   R5 ;DO ALL ADDRESSES -1
      MOV   R3,    (R2)+ ;PUT IUT INTO FIRST LOC OF BLOCK.
2$:   MOV   R4,    (R2) ;PUT 'RTS R5' FOLLOWING IUT.
      JSR   R5,    -(R2) ;GO EXECUTE THE IUT.
      MOV   (R2)+, R1 ;GET THE DATA FROM THE MEM ADR UNDER TEST.
      CMP   R0,    R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
      BEQ   65$,   ;BRANCH OVER ERROR CALL IF GOOD DATA.
64$:  JSR   PC,    SPRNT3 ;SET UP VALUES FOR ERROR PRINTING.
      JSR   PC,    $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
      .WORD 21 ;ERROR TYPE CODE.
65$:  MOV   R3,    (R2)+ ;PUT THE IUT INTO THE NEXT LOCATION.
      SOB  R5,    2$ ;BRANCH IF MORE IN CURRENT BLOCK.
      JSR  PC,    MMUP ;FIND NEXT BLOCK AND LOOP TO 1$.
    
```


2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474

011264		
011264	004567	006064
011270	012703	110412
011274	012704	000205
011300	012700	110605
011304	004467	001470
011310	005305	
011312	010322	
011314	010412	
011316	004542	
011320	012201	
011322	020001	
011324	001405	
011326	004767	005266
011332	004767	006330
011336	000021	
011340		
011340	010322	
011342	077514	
011344	004767	001642

```

*****
*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 LOCATION      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.
*****

```

```

TST16:
DIDBL: JSR    R5,    $SCOPE ;GO TO SCOPE ROUTINE.
        MOV    #110412,R3 ;GET 'MOVB R4,(R2)' INSTRUCTION (IUT).
        MOV    #205,  R4  ;GET 'RTS R5'
        MOV    #110605,R0 ;SET UP S/B DATA AFTER EXECUTION.
        JSR    R4,    INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$:     DEC    R5        ;DO ALL ADDRESSES -1
        MOV    R3,    (R2)+ ;PUT IUT INTO FIRST LOC OF BLOCK.
2$:     MOV    R4,    (R2)  ;PUT 'RTS R5' FOLLOWING IUT.
        JSR    R5,    -(R2) ;GO EXECUTE THE IUT.
        MOV    (R2)+,  R1    ;GET THE DATA FROM THE MEM ADR UNDER TEST.
        CMP    R0,    R1    ;COMPARE THE CHECK WORD WITH THE DATA READ.
        BEQ    65$      ;BRANCH OVER ERROR CALL IF GOOD DATA.
64$:    JSR    PC,    SPRNT3 ;SET UP VALUES FOR ERROR PRINTING.
        JSR    PC,    $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
        .WORD 21          ;ERROR TYPE CODE.
65$:    MOV    R3,    (R2)+ ;PUT THE IUT INTO THE NEXT LOCATION.
        SOB   R5,    2$    ;BRANCH IF MORE IN CURRENT BLOCK.
        JSR   PC,    MMUP  ;FIND NEXT BLOCK AND LOOP TO 1$.

```


2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567

011440			
011440	004567	005710	
011444	012703	005412	
011450	012704	000205	
011454	012700	172366	
011460	004467	001314	
011464	005305		
011466	010322		
011470	010412		
011472	004542		
011474	012201		
011476	020001		
011500	001405		
011502	004767	005112	
011506	004767	006154	
011512	000021		
011514			
011514	010322		
011516	077514		
011520	004767	001466	

```

*****
*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 LOCATION      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.

```

```

*****
TST20:
DIPDO: JSR      R5,      $SCOPE ;GO TO SCOPE ROUTINE.
        MOV      #005412,R3 ;GET 'NEG (R2)' INSTRUCTION (IUT).
        MOV      #205,  R4  ;GET 'RTS R5'
        MOV      #172366,R0 ;SET UP S/B DATA AFTER EXECUTION.
        JSR      R4,      INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$:     DEC      R5 ;DO ALL ADDRESSES -1
        MOV      R3,      (R2)+ ;PUT IUT INTO FIRST LOC OF BLOCK.
2$:     MOV      R4,      (R2) ;PUT 'RTS R5' FOLLOWING IUT.
        JSR      R5,      -(R2) ;GO EXECUTE THE IUT.
        MOV      (R2)+,  R1 ;GET THE DATA FROM THE MEM ADR UNDER TEST.
        CMP      R0,      R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
        BEQ      65$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
64$:    JSR      PC,      SPRNT3 ;SET UP VALUES FOR ERROR PRINTING.
        JSR      PC,      $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
        .WORD    21 ;ERROR TYPE CODE.
65$:    MOV      R3,      (R2)+ ;PUT THE IUT INTO THE NEXT LOCATION.
        SOB     R5,      2$ ;BRANCH IF MORE IN CURRENT BLOCK.
        JSR     PC,      MMUP ;FIND NEXT BLOCK AND LOOP TO 1$.

```

2568
 2569
 2570
 2571
 2572
 2573
 2574
 2575
 2576
 2577
 2578
 2579
 2580
 2581
 2582
 2583
 2584
 2585
 2586
 2587
 2588
 2589
 2590
 2591
 2592
 2593
 2594
 2595
 2596
 2597
 2598
 2599
 2600
 2601
 2602
 2603
 2604
 2605
 2606
 2607
 2608
 2609
 2610
 2611
 2612

011524
 011524 004567 005624
 011530 012703 142242
 011534 012704 000205
 011540 012700 142000
 011544 004467 001230
 011550 005305
 011552 010322
 011554 010412
 011556 004542
 011560 012201
 011562 020001
 011564 001405
 011566 004767 005026
 011572 004767 006070
 011576 000021
 011600
 011600 010322
 011602 077514
 011604 004767 001402

```

*****
*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 LOCATION      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.
*****
    
```

```

TST21:
DPDBL: JSR      R5,      $SCOPE      ;GO TO SCOPE ROUTINE.
        MOV      #142242, R3        ;GET BICB (R2)+, -(R2)' INSTRUCTION (IUT).
        MOV      #205,   R4         ;GET 'RTS R5'
        MOV      #142000, R0        ;SET UP S/B DATA AFTER EXECUTION.
        JSR      R4,      INITMM    ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$:     DEC      R5              ;DO ALL ADDRESSES -1
        MOV      R3,      (R2)+     ;PUT IUT INTO FIRST LOC OF BLOCK.
2$:     MOV      R4,      (R2)      ;PUT 'RTS R5' FOLLOWING IUT.
        JSR      R5,      -(R2)     ;GO EXECUTE THE IUT.
        MOV      (R2)+,   R1        ;GET THE DATA FROM THE MEM ADR UNDER TEST.
        CMP      R0,      R1        ;COMPARE THE CHECK WORD WITH THE DATA READ.
        BEQ      65$          ;BRANCH OVER ERROR CALL IF GOOD DATA.
64$:    JSR      PC,      SPRNT3    ;SET UP VALUES FOR ERROR PRINTING.
        JSR      PC,      $ERROR    ;*** ERROR *** (GO TYPE A MESSAGE)
        .WORD    21              ;ERROR TYPE CODE.
65$:    MOV      R3,      (R2)+     ;PUT THE IUT INTO THE NEXT LOCATION.
        SOB     R5,      2$         ;BRANCH IF MORE IN CURRENT BLOCK.
        JSR     PC,      MMUP      ;FIND NEXT BLOCK AND LOOP TO 1$.
    
```


2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658

011610
011610 004567 005540
011614 012703 152212
011620 012704 000205
011624 012700 157212
011630 004467 001144
011634 005305
011636 010322
011640 010412
011642 004542
011644 005302
011646 012201
011650 020001
011652 001405
011654 004767 004740
011660 004767 006002
011664 000021
011666
011666 010322
011670 077515
011672 004767 001314

```

*****
*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 LOCATION      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.
*****

```

```

1ST22:
DPDBH: JSR   R5,   $SCOPE ;GO TO SCOPE ROUTINE.
        MOV   #152212,R3 ;GET 'BISB (R2)+,(R2)' INSTRUCTION (IUT).
        MOV   #205,  R4  ;GET 'RTS R5'
        MOV   #157212,R0 ;SET UP S/B DATA AFTER EXECUTION.
        JSR   R4,   INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$:     DEC   R5        ;DO ALL ADDRESSES -1
        MOV   R3,    (R2)+ ;PUT IUT INTO FIRST LOC OF BLOCK.
2$:     MOV   R4,    (R2)  ;PUT 'RTS R5' FOLLOWING IUT.
        JSR   R5,    -(R2) ;GO EXECUTE THE IUT.
        DEC   R2        ;RESET R2 TO POINT TO IUT.
        MOV   (R2)+, R1   ;GET THE DATA FROM THE MEM ADR UNDER TEST.
        CMP   R0,    R1   ;COMPARE THE CHECK WORD WITH THE DATA READ.
        BEQ   6$,    ;BRANCH OVER ERROR CALL IF GOOD DATA.
64$:    JSR   ,      SPRNT3 ;SET UP VALUES FOR ERROR PRINTING.
        JSR   PC,    $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
        .WORD 21         ;ERROR TYPE CODE.
65$:    MOV   R3,    (R2)+ ;PUT THE IUT INTO THE NEXT LOCATION.
        SOB  R5,    2$    ;BRANCH IF MORE IN CURRENT BLOCK.
        JSR  PC,    MMUP   ;FIND NEXT BLOCK AND LOOP TO 1$.

```

2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684 011676
2685 011676 004567 005452
2686 011702 004467 001072
2687 011706 010567 170060
2688 011712 010267 170052
2689 011716 005003
2690 011720 012700 000377
2691 011724 010022
2692 011726 077502
2693 011730 016705 170036
2694 011734 014201
2695 011736 020001
2696 011740 001405
2697 011742 004767 004656
2698 011746 004767 005714
2699 011752 000010
2700 011754
2701 011754 000300
2702 011756 010012
2703 011760 011201
2704 011762 020001
2705 011764 001405
2706 011766 004767 004632
2707 011772 004767 005670
2708 011776 000010
2709 012000
2710 012000 000300
2711 012002 005703
2712 012004 001403
2713 012006 020327 000003
2714 012012 001011

```

.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.
*****
TST23:
      JSR      R5,      $SCOPE ;GO TO SCOPE ROUTINE.
      JSR      R4,      INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
100$:  MOV      R5,TEMP1 ;STORE TEST WORD COUNTER
      MOV      R2,TEMP ;SAVE BANK STARTING ADDRESS
1$:    CLR      R3 ;CLEAR PASS COUNTER
      MOV      #000377,R0 ;SETUP TO WRITE PATTERN
2$:    MOV      R0,(R2)+ ;WRITE PATTERN
      SOB      R5,2$ ;END OF 8K?
      MOV      TEMP1,R5 ;RESTORE TEST WORD COUNTER FOR NEXT PASS
3$:    MOV      -(R2),R1 ;GET DATA WRITEN
      CMP      R0,      R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
      BEQ      65$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
64$:  JSR      PC,      SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
      JSR      PC,      $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
      .WORD    10 ;ERROR TYPE CODE.
65$:  4$:    SWAB     R0 ;SWAP BYTES OF DATA
      MOV      R0,(R2) ;WRITE SWAPPED WORD
      MOV      (R2),R1 ;GET DATA WRITEN
      CMP      R0,      R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
      BEQ      67$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
66$:  JSR      PC,      SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
      JSR      PC,      $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
      .WORD    10 ;ERROR TYPE CODE.
67$:  SWAB     R0 ;PUT DATA BACK TO ORINGINAL
      TST      R3 ;IF ON PASS 0 OR PASS 3
      BEQ      5$ ;WE ARE ADDRESSING DOWN
      CMP      R3,#3 ;IF ON PASS 1 OR 2 GO TO
      BNE     6$ ;UPWARD

```



```

2715 012014 077531      5$: SOB      R5,3$      ;DONE A PASS? IF NOT THEN 3$
2716 012016 005203      INC      R3        ;IF YES INCREMENT PASS COUNTER
2717 012020 022703 000004  CMP      #4,R3     ;ARE WE DONE ALL PASSES FOR THIS 8K?
2718 012024 001433      BEQ      9$        ;IF YES BRANCH
2719 012026 016705 167740  MOV      TEMP1,R5  ;RESTORE TEST COUNTER FOR NEXT PASS.
2720 012032 000300      SWAB    R0        ;ELSE SET UP NEW READ WORD
2721 012034 000404      BR      7$        ;GO TO START OF ADDRESS UP
2722 012036 062702 000002  6$: ADD      #2,R2   ;UPDATE TO NEXT ADDRESS
2723 012042 005305      DEC      R5        ;DONE A PASS?
2724 012044 001411      BEQ      8$        ;IF YES BRANCH
2725 012046 011201      7$: MOV      (R2),R1 ;GET DATA WRITTEN
2726 012050 020001      CMP      R0,      R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
2727 012052 001405      BEQ      69$       ;BRANCH OVER ERROR CALL IF GOOD DATA.
2728 012054 004767 004544  68$: JSR      PC,      SPRINT2 ;SET UP VALUES FOR ERROR PRINTING.
2729 012060 004767 005602  JSR      PC,      $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2730 012064 000010      .WORD   10        ;ERROR TYPE CODE.
2731 012066
2732 012066 000732      69$: BR      4$
2733 012070 016705 167676  8$: MOV      TEMP1,R5 ;RESTORE TEST WORD COUNTER FOR NEXT PASS.
2734 012074 005203      INC      R3        ;INCREMENT PASS COUNTER
2735 012076 000300      SWAB    R0        ;SET UP NEW READ WORD
2736 012100 020327 000002  CMP      R3,#2     ;ADDRESSING UP?
2737 012104 001313      BNE     3$        ;IF NO GO TO DOWN SEQUENCE
2738 012106 016702 167656  MOV      TEMP,R2   ;IF YES RESET ADDRESS TO START
2739 012112 000755      BR      7$        ;GO TO UP SEQUENCE
2740 012114 062702 040000  9$: ADD      #40000,R2 ;TESTING ENDS WITH R2 = BOTTOM OF BANK
2741
2742 012120 042702 001000  BIC      #1000,R2  ;ADD 8K FOR TEST START OF NEXT BANK
2743
2744
2745 012124 004767 001062  JSR      PC,MMUP   ;CLEAR POSSIBLE FIRST STARTING
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761 012130
2762 012130 004567 005220  JSR      R5,      $SCOPE ;GO TO SCOPE ROUTINE.
2763 012134 004467 000640  JSR      R4,      INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2764 012140 012700 125252  MOV      #125252,R0 ;SETUP DATA PATTERN
2765 012144 010022      MOV      R0,      (R2)+ ;WRITE A WORD
2766 012146 005100      COM     R0        ;COMPLEMENT DATA
2767 012150 077503      SOB     R5,      1$    ;BRANCH IF MORE IN CURRENT BLOCK
2768 012152 004767 001034  JSR      PC,      MMUP  ;FIND NEXT BLOCK AND LOOP TO 1$.
2769 012156 005003      CLR     R3        ;SET UP COUNTER FOR STALL
2770 012160 012704 000046  MOV      #46,      R4   ;DO LOOP 46 TIMES OR 2 SEC. TOTAL.
    
```

```

*****
*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.
*****
    
```

```

TST24:
1$: JSR      R5,      $SCOPE ;GO TO SCOPE ROUTINE.
    JSR      R4,      INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
    MOV      #125252,R0 ;SETUP DATA PATTERN
    MOV      R0,      (R2)+ ;WRITE A WORD
    COM     R0        ;COMPLEMENT DATA
    SOB     R5,      1$    ;BRANCH IF MORE IN CURRENT BLOCK
    JSR      PC,      MMUP  ;FIND NEXT BLOCK AND LOOP TO 1$.
    CLR     R3        ;SET UP COUNTER FOR STALL
    MOV      #46,      R4   ;DO LOOP 46 TIMES OR 2 SEC. TOTAL.
    
```

```

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

```



```
2818 .SBTTL DONE: RELOCATE PROGRAM AND REPEAT ALL TESTS.  
2819 012340 DONE:  
2820 012340 004567 005010 JSR R5, $SCOPE ;GO TO SCOPE ROUTINE.  
2821 012344 005067 166610 TST32: CLR $TIMES ;RESET ITERATION COUNTER FOR RESTARTING TEST.  
2822 012350 105067 166516 CLR $TSTNM ;RESET TEST NUMBER.  
2823 012354 032777 000200 166546 BIT #SW07, @SWR ;CHECK FOR INHIBIT RELOCATION SWITCH.  
2824 012362 001037 BNE $EOP ;SKIP RELOCATION IF SWITCH SET.  
2825 012364 105767 166624 TSTB $ENV ;CHECK FOR APT11  
2826 012370 001403 BEQ 1$ ;BRANCH IF NOT APT11  
2827 012372 005767 166604 TST $PASS ;CHECK FOR 1ST PASS  
2828 012376 001026 BNE 6$ ;IF APT,DO NOT RELOCATE AFTER FIRST PASS  
2829 012400 026767 165436 165440 1$: CMP 42,46 ;CHECK FOR ACT11  
2830 012406 001003 BNE 2$ ;BRANCH IF NOT ACT11  
2831 012410 005767 166566 TST $PASS ;CHECK FOR FIRST PASS  
2832 012414 001417 BEQ 6$ ;IF ACT,DO NOT RELOCATE ON FIRST PASS  
2833 012416 032767 000001 166110 2$: BIT #BIT0,PRGMAP ;CHECK IF PROGRAM IN FIRST 8K  
2834 012424 001404 BEQ 4$ ;IF NOT IN FIRST 8K,RELOC TOP TO BOTTOM  
2835 ;MUST BE XXDP OR STANDALONE  
2836 012426 004767 002140 JSR PC, RELTOP ;RELOCATE PROGRAM TO TOP OF MEMORY.  
2837 012432 000167 174224 3$: JMP START1 ;LOOP BACK AND RUN ALL TESTS AGAIN.  
2838  
2839 012436 004767 002444 4$: JSR PC, RELO ;RELOCATE PROGRAM BACK TO FIRST 8K.  
2840 012442 005767 165374 TST 42 ;TEST FOR XXDP  
2841 012446 001402 BEQ 6$ ;IF NOT RUNNING UNDER MON. DONT  
2842 012450 004767 002632 5$: JSR PC, RESLDR ;RESTORE LOADERS.  
2843 012454 6$:  
2844 012454 004567 007114 JSR R5, $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.  
2845 012460 001171 .WORD $CRLF ;ADDRESS OF MESSAGE TO BE TYPED
```

2846
2847
2848
2849
2850
2851
2852
2853
2854
2855 012462
2856 012462 000240
2857 012464 005067 166470
2858 012470 005267 166506
2859 012474 042767 100000 166500
2860 012502 005327
2861 012504 000001
2862 012506 003041
2863 012510 012737
2864 012512 000001
2865 012514 012504
2866 012516 004567 007052
2867 012522 012616
2868 012524 016746 166452
2869
2870
2871 012530 106746
2872 012532 105066 000001
2873
2874 012536 004767 010026
2875 012542 004567 007026
2876 012546 012633
2877 012550
2878
2879 012550 016700 165266
2880 012554 001416
2881 012556 000005
2882 012560 004710
2883 012562 000240
2884 012564 000240
2885 012566 000240
2886 012570 023737 000042 000046
2887 012576 001405
2888 012600 105737 001214
2889 012604 001002
2890 012606 004767 002562
2891 012612
2892 012612 000167 174044
2893 012616 005015 047105 020104
2894 012624 040520 051523 021440
2895 012632 000
2896 012633 377 377 000
2897
2898
2899
2900
2901

```
::*****  
.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  
$EOP:  
NOP  
CLR $TIMES ;;ZERO THE NUMBER OF ITERATIONS  
INC $PASS ;;INCREMENT THE PASS NUMBER  
BIC #100000,$PASS ;;DON'T ALLOW A NEG. NUMBER  
DEC (PC)+ ;;LOOP?  
$EOPCT: .WORD 1  
BGT $DOAGN ;;YES  
MOV (PC)+,@(PC)+ ;;RESTORE COUNTER  
$ENDCT: .WORD 1  
JSR R5,$$PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.  
;.WORD $ENDMG ;ADDRESS OF MESSAGE TO BE TYPED  
MOV $PASS,-(SP) ;;SAVE $PASS FOR TYPEOUT  
:* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPDS ROUTINE  
:* WIHTOUT USING A 'TRAP' INSTRUCTION AS CALLED FOR BY **SYSMAC**.  
MFPS -(SP) ;PUT THE PROCESSOR STATUS ON THE STACK  
CLRB 1(SP) ;HIGH BYTE CLEARED TO INSURE KERNEL MODE  
;ON PSW RETURN.  
JSR PC,$$TYPDS ;GO TO THE SUBROUTINE  
JSR R5,$$PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.  
;.WORD $ENULL ;ADDRESS OF MESSAGE TO BE TYPED  
$GET42:  
MOV 42,R0 ;;GET MONITOR ADDRESS  
BEQ $DOAGN ;;BRANCH IF NO MONITOR  
RESET ;;CLEAR THE WORLD  
$ENDAD: JSR PC,(R0) ;;GO TO MONITOR  
NOP ;;SAVE ROOM  
NOP ;;FOR  
NOP ;;ACT11  
CMP @#42,@#46 ;;ARE WE UNDER ACT11 OR XXDP  
BEQ $DOAGN ;;IF ACT11 THEN RESTART  
TSTB @#$ENV ;;CHECK FOR APT11  
BNE $DOAGN ;;IF APT11 THEN RESTART  
JSR PC,SAVLDR ;;IF XXDP FIRST SAVE MONITOR  
$DOAGN: JMP START1 ;;RETURN*****  
$ENDMG: .ASCIZ <15><12>/END PASS #/  
$ENULL: .BYTE -1,-1,0 ;;NULL CHARACTER STRING  
.SBTTL SUBROUTINE AND TRAP ROUTINE SECTION.  
.SBTTL MEMORY MANAGEMENT AND ADDRESSING SUBROUTINES.  
:* *****  
:* SET UP ALL THE MEM MGMT REGISTERS FOR NORMAL OPERATION.  
:* THE PROGRAM IS POINTED TO BY PARS 0 AND 1.
```


;* THE MEMORY UNDER TEST IS POINTED TO BY PARS 2 AND 3.
;* THE DEVICE ADDRESS AREA IS POINTED TO BY PAR 7.
;* PARS 4, 5, AND 6 ARE UNUSED.
:*****

2902
2903
2904
2905
2906 012636
2907 012636 012737 077406 172300
2908 012644 012737 077406 172302
2909 012652 012737 077406 172304
2910 012660 012737 077406 172306
2911 012666 005037 172310
2912 012672 005037 172312
2913 012676 005037 172314
2914 012702 012737 077406 172316
2915 012710 005037 172340
2916 012714 012737 000200 172342
2917 012722 005037 172344
2918 012726 005037 172346
2919 012732 005037 172350
2920 012736 005037 172352
2921 012742 005037 172354
2922 012746 012737 177600 172356
2923 012754 005767 165556
2924 012760 100003
2925 012762 052737 000020 172516
2926 012770 012737 000001 177572
2927 012776 000207

```
MMINIT:
MOV #200-1*400+UP+RW,@#KIPDR0 ;SET KIPDR0 = RW UP 200 BLOCKS
MOV #200-1*400+UP+RW,@#KIPDR1 ;SET KIPDR1 = RW UP 200 BLOCKS
MOV #200-1*400+UP+RW,@#KIPDR2 ;SET KIPDR2 = RW UP 200 BLOCKS
MOV #200-1*400+UP+RW,@#KIPDR3 ;SET KIPDR3 = RW UP 200 BLOCKS
CLR @#KIPDR4
CLR @#KIPDR5
CLR @#KIPDR6
MOV #200-1*400+UP+RW,@#KIPDR7 ;SET KIPDR7 = RW UP 200 BLOCKS
CLR @#KIPAR0 ;MAP PAR0 INTO BANK0
MOV #200,@#KIPAR1 ;MAP PAR1 INTO BANK1
CLR @#KIPAR2 ;MAP PAR2 INTO BANK0
CLR @#KIPAR3
CLR @#KIPAR4
CLR @#KIPAR5
CLR @#KIPAR6
MOV #177600,@#KIPAR7 ;MAP PAR7 INTO I/O BANK
TST MMAVA ;IF 22 BIT ADR NOT AVAILABLE
BPL 1$ ; THEN BRANCH
BIS #BIT4,@#SR3 ; ELSE ENABLE 22 BIT ADDRESSING
1$: MOV #1,@#SRO ;ENABLE MEMORY MANAGEMENT
RTS PC ;RETURN
```

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

2928
2929
2930
2931
2932
2933
2934 013000
2935 013000 010046
2936 013002 010146
2937 013004 012767 100000 166766
2938 013012 005002
2939 013014 005767 165516
2940 013020 001410
2941 013022 012700 000020
2942 013026 016701 166762
2943 013032 005021
2944 013034 077002
2945 013036 005037 172344
2946 013042 012767 013204 166704
2947 013050 066767 165456 166676
2948 013056 012601
2949 013060 012600
2950 013062 004767 000124
2951 013066 004767 004574
2952
2953 013072 000007
2954 013074 000000
2955
2956
2957 013076

```
INITMM:
MOV R0,-(SP) ;;PUSH R0 ON STACK
MOV R1,-(SP) ;;PUSH R1 ON STACK
MOV #BIT15,PLUS1
CLR R2 ;INITIALIZE ADDRESS
TST MMAVA
BEQ 2$
MOV #20,R0
MOV .BITPT,R1
1$: CLR (R1)+
SOB R0,1$
CLR @#KIPAR2
2$: MOV #INITEX,MMORE
ADD RELOCF,MMORE
MOV (SP)+,R1 ;;POP STACK INTO R1
MOV (SP)+,R0 ;;POP STACK INTO R0
JSR PC,MMUP
JSR PC,$ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
:FATAL ERROR
WORD 7 ;ERROR TYPE CODE.
HALT ;FATAL ERROR!!! NO MEM INDICATED IN MEMMAP ABOVE 8K!
```

INITDN:

```

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


3014	013310	012604				MOV	(SP)+,R4	::POP STACK INTO R4
3015	013312	012603				MOV	(SP)+,R3	::POP STACK INTO R3
3016	013314	012601				MOV	(SP)+,R1	::POP STACK INTO R1
3017	013316	012600				MOV	(SP)+,R0	::POP STACK INTO R0
3018	013320	000510				BR	12\$:DONE,READY TO EXIT
3019	013322	016703	166466		101\$:	MOV	.BITPT,R3	
3020	013326	016704	166456			MOV	.TSTMAP,R4	
3021	013332	012700	000020			MOV	#20,R0	
3022	013336	032324			3\$:	BIT	(R3)+,(R4)+	
3023	013340	001002				BNE	4\$	
3024	013342	077003				SOB	R0,3\$	
3025	013344	000745				BR	1\$	
3026	013346	013737	172344	172346	4\$:	MOV	@#KIPAR2,@#KIPAR3	
3027	013354	062737	000200	172346		ADD	#200,@#KIPAR3	
3028	013362	012604				MOV	(SP)+,R4	::POP STACK INTO R4
3029	013364	012603				MOV	(SP)+,R3	::POP STACK INTO R3
3030	013366	012601				MOV	(SP)+,R1	::POP STACK INTO R1
3031	013370	012600				MOV	(SP)+,R0	::POP STACK INTO R0
3032	013372	005767	165140			TST	MMAVA	
3033	013376	100005				BPL	5\$	
3034	013400	032767	100000	166344		BIT	#BIT15,BITPT+36	
3035	013406	001005				BNE	6\$	
3036	013410	000436				BR	9\$	
3037	013412	032767	100000	166274	5\$:	BIT	#BIT15,BITPT	
3038	013420	001432				BEQ	9\$	
3039	013422	012705	010000		6\$:	MOV	#10000,R5	
3040	013426	000427				BR	9\$	
3041	013430	006367	166344		7\$:	ASL	PLUS1	
3042	013434	006167	166254			ROL	BITPT	
3043	013440	103440				BCS	12\$	
3044	013442	036767	166246	166104		BIT	BITPT,TSTMAP	
3045	013450	001003				BNE	8\$	
3046	013452	062702	040000			ADD	#40000,R2	
3047	013456	000764				BR	7\$	
3048	013460	032767	000010	166226	8\$:	BIT	#BIT3,BITPT	
3049	013466	001407				BEQ	9\$	
3050	013470	012705	010000			MOV	#10000,R5	
3051	013474	005767	166274			TST	FLG30K	
3052	013500	001402				BEQ	9\$	
3053	013502	062705	004000			ADD	#4000,R5	
3054	013506	032767	000001	166200	9\$:	BIT	#BIT0,BITPT	
3055	013514	001407				BEQ	10\$	
3056	013516	066702	166240			ADD	FSTADR,R2	
3057	013522	005767	166234			TST	FSTADR	:FSTADR = STARTING ADDRESS
3058								: 0 OR 1000 IF VECTOR AREA IS
3059								: PROTECTED.
3060	013526	001402				BEQ	10\$	
3061	013530	162705	000400			SUB	#400,R5	:ADJUST ADDR COUNTER TO PROTECT 0-1000
3062	013534	016716	166214		10\$:	MOV	MORE,(SP)	
3063	013540	000207			11\$:	RTS	PC	
3064								:* BEFORE FINAL EXIT, CHECK FOR ANY NON-TRAP PARITY ERRORS.
3065	013542	005767	167622		12\$:	TST	MPRX	:CHECK FOR ANY PARITY REGISTERS PRESENT.
3066	013546	001402				BEQ	13\$:BR IF NONE.
3067	013550	004767	002106			JSR	PC,CKPMER	:CHECK FOR PARITY MEMORY ERRORS.
3068	013554	000207			13\$:	RTS	PC	:STRAIGHT RETURN.
3069								

```

3070
3071
3072
3073
3074
3075
3076
3077 013556 012705 020000
3078
3079
3080 013562 106746
3081 013564 105066 000001
3082
3083 013570 004767 004634
3084 013574 005767 164736
3085 013600 001502
3086 013602 010046
3087 013604 010146
3088 013606 010346
3089 013610 010446
3090 013612 162737 000400 172344 1$:
3091 013620 010700 166170
3092 013624 062700 000040
3093 013630 012703 000020
3094 013634 006267 166140
3095 013640 006040 2$:
3096 013642 077302
3097 013644 103005
3098 013646 012604
3099 013650 012603
3100 013652 012601
3101 013654 012600
3102 013656 000517
3103 013660 016700 166130 100$:
3104 013664 062700 000040
3105 013670 016701 166114
3106 013674 062701 000040
3107 013700 012703 000020
3108 013704 034041 3$:
3109 013706 001002
3110 013710 077303
3111 013712 000737
3112 013714 013737 172344 172346 4$:
3113 013722 062737 000200 172346
3114 013730 012702 100000
3115 013734 012604
3116 013736 012603
3117 013740 012601
3118 013742 012600
3119 013744 005767 164566
3120 013750 100005
3121 013752 032767 100000 165772
3122 013760 001005
3123 013762 000442
3124 013764 032767 100000 165722 5$:
3125 013772 001436

```

```

*****
* MEMORY DOWNWARDS ADDRESSING SUBROUTINE.
* FINDS NEXT LOWER 8K BANK AND UPDATES POINTERS.
* GOES TO ADDRESS IN 'MMORE' IF MORE BANKS.
* DOES STRAIGHT EXIT WHEN ALL MEMORY HAS BEEN DONE.
*****

```

```

MMDOWN: 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**.
MFPS -(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
CLRB 1(SP) ;HIGH BYTE CLEARED TO INSURE KERNEL MODE
;ON PSW RETURN.
JSR PC, $CKSWR ;GO TO THE SUBROUTINE
TST MMAVA ;CHECK FOR MEMORY MANAGEMENT
BEQ 7$ ;IF MMAVA = 0 THEN BR TO NON KT MODE
MOV R0,-(SP) ;:PUSH R0 ON STACK
MOV R1,-(SP) ;:PUSH R1 ON STACK
MOV R3,-(SP) ;:PUSH R3 ON STACK
MOV R4,-(SP) ;:PUSH R4 ON STACK
SUB #400,@#KIPAR2 ;DOWN DATE TO NEXT 8K BANK
MOV .BITPT,R0
ADD #40,R0 ;SET UP TO MOVE POINTER
MOV #20,R3 ;ONE POSITION THRU 20 WORD TABLE
ASR PLUS1 ;GET INITIAL VALUE OF BIT
ROR -(R0) ;MOVE THE POINTER
SOB R3,2$
BCC 100$ ;CONTINUE TESTING UNTIL CARRY BIT SET
MOV (SP)+,R4 ;:POP STACK INTO R4
MOV (SP)+,R3 ;:POP STACK INTO R3
MOV (SP)+,R1 ;:POP STACK INTO R1
MOV (SP)+,R0 ;:POP STACK INTO R0
BR 11$ ;TESTING DONE,READY TO EXIT
MOV .BITPT,R0 ;THEN WE HAVE TESTED LAST BANK
ADD #40,R0 ;IF NO UNDERFLOW SET UP TO
MOV .TSTMAP,R1 ;SEE IF BANK SHOULD BE TESTED
ADD #40,R1
MOV #20,R3
BIT -(R0),-(R1) ;CHECK IF BANK IS UNDER TEST
BNE 4$ ;IF YES GO SET UP PARS
SOB R3,3$ ;IF NO CHECK ALL POSSIBLES
BR 1$ ;IF NOT GO CHECK NEXT LOWER 8K
MOV @#KIPAR2,@#KIPAR3 ;MAKE PAR2 AND PAR3 POINT TO SAME
ADD #200,@#KIPAR3 ;MAKE KIPAR3 UPPER 4K
MOV #100000,R2 ;SET UP FIRST ADDRESS + 2
MOV (SP)+,R4 ;:POP STACK INTO R4
MOV (SP)+,R3 ;:POP STACK INTO R3
MOV (SP)+,R1 ;:POP STACK INTO R1
MOV (SP)+,R0 ;:POP STACK INTO R0
TST MMAVA ;CHECK FOR 22 BIT ADDRESSING
BPL 5$ ;IF NO,CHECK 18 BIT MODE
BIT #BIT15,BITPT+36 ;TESTING HIGHEST 22 BIT SEG.?
BNE 6$ ;IF YES,DO ONLY 4K SEGMENT
BR 9$ ;IF NO,DO 8K SEGMENT
BIT #BIT15,BITPT ;TESTING HIGHEST 18 BIT SEGMENT
BEQ 9$ ;IF NO ,DO 8K SEGMENT

```



```

3152 .SBTTL SUBROUTINES FOR ADDRESS AND WORSE CASE NOISE TESTS.
3153 :*****
3154 :* SUBROUTINE TO CALCULATE PHYSICAL ADDRESS AND PUT IT IN R0 (BOTTOM 16 BITS).
3155 :*****
3156 014120 005767 164412 PHYADR: TST MMAVA ;CHECK FOR MEM MGMT AVAILABLE
3157 014124 001433 BEQ 4$ ;BRANCH IF NO MEM MGMT
3158 014126 010146 MOV R1,-(SP) ;:PUSH R1 ON STACK
3159 014130 010346 MOV R3,-(SP) ;:PUSH R3 ON STACK
3160 014132 010446 MOV R4,-(SP) ;:PUSH R4 ON STACK
3161 014134 010546 MOV R5,-(SP) ;:PUSH R5 ON STACK
3162 014136 010005 1$: MOV R0,R5 ;SAVE FOR PHYSICAL ADDR. ADDITION
3163 014140 042700 017777 BIC #17777,R0 ;CLEAR VIRTUAL ADDR BITS
3164 014144 012701 000014 MOV #12.,R1 ;SET UP ROTATE COUNTER
3165 014150 006200 2$: ASR R0 ;TO SHIFT PAR POINTER
3166 014152 077102 SOB R1,2$ ;INTO BITS 0-3
3167 014154 012704 172340 MOV #KIPAR0,R4 ;STORE START OF PARS
3168 014160 060004 ADD R0,R4 ;USE PAR POINTER IN 0-3 AS INDEX
3169 014162 011403 MOV (R4),R3 ;STORE CONTENTS OF PROPER PAR
3170 014164 012701 000006 MOV #6,R1 ;SET UP COUNTER TO
3171 014170 006303 3$: ASL R3 ;CONVERT PAR CONTENTS
3172 014172 077102 SOB R1,3$ ;TO A PHYSICAL 'K' ADDRESS
3173 014174 010500 MOV R5,R0 ;STORE VIRTUAL ADDRESS
3174 014176 042700 160000 BIC #160000,R0 ;CLEAR PAR POINTER
3175 014202 060300 ADD R3,R0 ;ADD PHYSICAL 'K' ADDRESS
3176 ;TO OBTAIN 16 BIT PHYSICAL ADDRESS
3177 014204 012605 MOV (SP)+,R5 ;:POP STACK INTO R5
3178 014206 012604 MOV (SP)+,R4 ;:POP STACK INTO R4
3179 014210 012603 MOV (SP)+,R3 ;:POP STACK INTO R3
3180 014212 012601 MOV (SP)+,R1 ;:POP STACK INTO R1
3181 014214 000207 4$: RTS PC ;EXIT WITH PHYSICAL ADDR IN R0
3182 :*****
3183 :* SUBROUTINE TO PUT BANK NUMBER INTO R0.
3184 :*****
3185 BANKNO: CLR R0 ;INIT R0
3186 014216 005000 MOV R1,-(SP) ;:PUSH R1 ON STACK
3187 014220 010146 MOV R2,-(SP) ;:PUSH R2 ON STACK
3188 014222 010246 MOV R3,-(SP) ;:PUSH R3 ON STACK
3189 014224 010346 MOV R4,-(SP) ;:PUSH R4 ON STACK
3190 014226 010446 MOV .BITPT,R1 ;START OF BIT POINTER TABLE
3191 014230 016701 165560 MOV #20,R3 ;SET UP BITPT TABLE COUNTER
3192 014234 012703 000020 1$: MOV #20,R2 ;SET UP ROTATE COUNTER FOR 1 WORD
3193 014240 012702 000020 MOV (R1)+,R4 ;GET WORD TO ROTATE BIT
3194 014244 012104 2$: ASR R4 ;ROTATE BITPT WORD TO FIND BANK
3195 014246 006204 BCS 4$ ;IF CARRY SET ,THEN BANK FOUND
3196 014250 103410 INCB R0 ;COUNT BANKS.
3197 014252 105200 BNE 3$ ;BR IF NOT OVERFLOW.
3198 014254 001004 JSR PC,$ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
3199 014256 004767 003404 ;FATAL ERROR
3200 .WORD 7 ;ERROR TYPE CODE.
3201 014262 000007 HALT ;FATAL ERROR!!! NO POINTER FOUND.
3202 014264 000000 3$: SOB R2,2$ ;IF NOT FINISHED ROTATING WORD
3203 014266 077211 SOB R3,1$ ;GO BACK AND ROTATE AGAIN
3204 ;IF NOT FINISHED BITPT TABLE
3205 014270 077315 ;THEN ROTATE THROUGH NEXT WORD.
3206
3207 014272 4$:

```


3208 014272 012604
 3209 014274 012603
 3210 014276 012602
 3211 014300 012601
 3212 014302 000207
 3213
 3214
 3215
 3216
 3217 014304
 3218 014304 004467 176470
 3219 014310 010022
 3220 014312 077502
 3221 014314 004767 176672
 3222 014320 000207
 3223
 3224
 3225
 3226
 3227 014322 106112
 3228 014324 106112
 3229 014326 106112
 3230 014330 106112
 3231 014332 106112
 3232 014334 106112
 3233 014336 106112
 3234 014340 106112
 3235 014342 106122
 3236 014344 106112
 3237 014346 106112
 3238 014350 106112
 3239 014352 106112
 3240 014354 106112
 3241 014356 106112
 3242 014360 106112
 3243 014362 106112
 3244 014364 106122
 3245 014366 000207
 3246
 3247
 3248
 3249
 3250 014370 012704 000020
 3251
 3252 014374 010022
 3253 014376 010022
 3254 014400 010022
 3255 014402 010022
 3256
 3257 014404 010322
 3258 014406 010322
 3259 014410 010322
 3260 014412 010322
 3261
 3262 014414 010022
 3263 014416 010022

```

MOV (SP)+,R4      ;;POP STACK INTO R4
MOV (SP)+,R3      ;;POP STACK INTO R3
MOV (SP)+,R2      ;;POP STACK INTO R2
MOV (SP)+,R1      ;;POP STACK INTO R1
RTS PC            ;;RETURN

*****
* SUBROUTINE TO WRITE THE CONSTANT IN R0 INTO ALL OF MEMORY.
*****
SETCON:
JSR R4, INITMM   ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$: MOV R0, (R2)+ ;MOV CONSTANT INTO MEMORY
SOB R5, 1$       ;BRANCH IF MORE IN CURRENT BLOCK
JSR PC, MMUP     ;FIND NEXT BLOCK AND LOOP TO 1$.
RTS PC          ;RETURN

*****
* ROUTINE TO ROTATE 'C' BIT THROUGH A MEMORY LOCATION.
*****
ROTATE: ROLB (R2)      ;(R2)=177776 OR 000001
        ROLB (R2)      ;(R2)=177775 OR 000002
        ROLB (R2)      ;(R2)=177773 OR 000004
        ROLB (R2)      ;(R2)=177767 OR 000010
        ROLB (R2)      ;(R2)=177757 OR 000020
        ROLB (R2)      ;(R2)=177737 OR 000040
        ROLB (R2)      ;(R2)=177677 OR 000100
        ROLB (R2)      ;(R2)=177777 OR 000000
        ROLB (R2)+     ;(R2)=177577 OR 000200
        ROLB (R2)      ;(R2)=177377 OR 000400
        ROLB (R2)      ;(R2)=176777 OR 001000
        ROLB (R2)      ;(R2)=175777 OR 002000
        ROLB (R2)      ;(R2)=173777 OR 004000
        ROLB (R2)      ;(R2)=167777 OR 010000
        ROLB (R2)      ;(R2)=157777 OR 020000
        ROLB (R2)      ;(R2)=137777 OR 040000
        ROLB (R2)      ;(R2)=077777 OR 100000
        ROLB (R2)+     ;(R2)=177777 OR 000000
RTS PC          ;RETURN

*****
* SUBROUTINE TO WRITE 3 XOR 9 PATTERN INTO 256. WORD BLOCK.
*****
W3X9: MOV #16.,R4    ;EACH LOOP WRITES 256. WORDS

2$: MOV R0,(R2)+
   MOV R0,(R2)+
   MOV R0,(R2)+
   MOV R0,(R2)+

   MOV R3,(R2)+
   MOV R3,(R2)+
   MOV R3,(R2)+
   MOV R3,(R2)+

   MOV R0,(R2)+
   MOV R0,(R2)+

```

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


3278
3279
3280
3281
3282 014450
3283 014450 010246
3284 014452 010346
3285 014454 010446
3286 014456 012502
3287 014460 012503
3288 014462 012704 020000
3289 014466 012223
3290 014470 005304
3291 014472 001375
3292 014474 012704 020000
3293 014500 024243
3294 014502 001417
3295 014504 011267 164404
3296 014510 011367 164402
3297 014514 010267 164370
3298 014520 010367 164366
3299 014524 004767 003136
3300 014530 000023
3301 014532 000000
3302 014534 162705 000004
3303 014540 000746
3304 014542 005304
3305 014544 001355
3306 014546 004567 005022
3307 014552 024526
3308
3309 014554 010346
3310 014556 004767 006526
3311 014562 012604
3312 014564 012603
3313 014566 012602
3314 014570 000205
3315
3316
3317
3318
3319
3320 014572 022767 000001 163734
3321 014600 001404
3322 014602 004767 003060
3323
3324 014606 000007
3325 014610 000000
3326 014612
3327 014612 010046
3328 014614 010146
3329 014616 005767 163714
3330 014622 001441
3331 014624 012737 007600 172346
3332 014632 012700 100000
3333 014636 162737 000400 172346

```
.SBTTL RELOCATION SUBROUTINES.  
*****  
* ROUTINE TO RELOCATE PROGRAM CODE  
*****  
RELOC:  
MOV R2,-(SP) ;;PUSH R2 ON STACK  
MOV R3,-(SP) ;;PUSH R3 ON STACK  
MOV R4,-(SP) ;;PUSH R4 ON STACK  
4$: MOV (R5)+, R2 ;;GET FIRST LOCATION.  
MOV (R5)+, R3 ;;GET FIRST LOCATION OF DESTINATION.  
MOV #20000, R4 ;;SET UP 8K COUNTER.  
1$: MOV (R2)+, (R3)+ ;;MOV THE DATA.  
DEC R4 ;;COUNT THE WORDS.  
BNE 1$ ;;BR IF MORE.  
MOV #20000, R4 ;;RESET THE COUNTER.  
2$: CMP -(R2), -(R3) ;;CHECK THE DATA JUST MOVED.  
BEQ 3$ ;;BR IF DATA OK.  
MOV (R2), $GDDAT ;;GET SOURCE DATA.  
MOV (R3), $BDDAT ;;GET DESTINATION DATA.  
MOV R2, $GDADR ;;GET SOURCE ADDRESS.  
MOV R3, $BDADR ;;GET DESTINATION ADDRESS.  
JSR PC, $ERROR ;;** ERROR ** (GO TYPE A MESSAGE)  
.WORD 23 ;;ERROR TYPE CODE.  
HALT ;;FATAL ERROR!!! RELOCATION FAILED.  
SUB #4, R5 ;;ADJUST RETURN POINTER.  
BR 4$ ;;GO BACK AND TRY AGAIN.  
3$: DEC R4 ;;COUNT WORDS.  
BNE 2$ ;;BR IF MORE.  
JSR R5, $PRINT ;;GO PRINT OUT THE FOLLOWING MESSAGE.  
.WORD PRELOC ;;ADDRESS OF MESSAGE TO BE TYPED  
;;'PROGRAM RELOCATED TO '  
MOV R3, -(SP) ;;PUT THE DATA ON THE STACK.  
JSR PC, $STYPAD ;;DETERMINE THE PHYSICAL ADDRESS AND TYPE IT.  
MOV (SP)+, R4 ;;POP STACK INTO R4  
MOV (SP)+, R3 ;;POP STACK INTO R3  
MOV (SP)+, R2 ;;POP STACK INTO R2  
RTS R5 ;;RETURN  
*****  
* SUBROUTINE TO MOVE PROGRAM FROM BOTTOM TO TOP 8K BANK OF 128KW MEMORY.  
* HIGHEST AVAILABLE BANK WITH KT WILL BE 700000 - 737776  
* HIGHEST AVAILABLE BANK NON-KT WILL BE 100000 - 137776  
*****  
RELTOP: CMP #1, PRGMAP ;;CHECK THAT THE PROGRAM IS NOW IN BANKS 0 AND 1.  
BEQ 1$ ;;BR IF OK  
JSR PC, $ERROR ;;** ERROR ** (GO TYPE A MESSAGE)  
;FATAL ERROR  
.WORD 7 ;;ERROR TYPE CODE.  
HALT ;;FATAL ERROR!!! PROG SHOULD BE IN BANKS 0 AND 1  
1$: MOV R0,-(SP) ;;PUSH R0 ON STACK  
MOV R1,-(SP) ;;PUSH R1 ON STACK  
TST MMAVA  
BEQ 10$  
MOV #7600, @#KIPAR3 ;;SET PAR TO TOP OF MEM  
MOV #BIT15, R0 ;;...HI 128K.  
2$: SUB #400, @#KIPAR3 ;;BACK DOWN ONE BANK.
```

```

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

```



```

3390
3391
3392
3393 015106 032767 000001 163420
3394 015114 001404
3395 015116 004767 002544
3396
3397 015122 000007
3398 015124 000000
3399 015126 005767 163404
3400 015132 001417
3401 015134 005037 172344
3402 015140 012737 000200 172346
3403 015146 004567 177276
3404 015152 000000
3405 015154 040000
3406 015156 005037 172340
3407 015162 012737 000200 172342
3408
3409 015170 000440
3410
3411 015172 016746 163334
3412 015176 011667 000004
3413 015202 004567 177242
3414 015206 000000
3415 015210 000000
3416 015212 161607
3417
3418 015214 161606
3419 015216 010046
3420 015220 012700 002002
3421
3422 015224 166620 000002
3423 015230 005720
3424 015232 001776
3425 015234 024027 177777
3426 015240 001371
3427 015242 012600
3428 015244 161637 000004
3429 015250 161637 000024
3430 015254 161637 000114
3431 015260 161667 163644
3432 015264 161667 163642
3433 015270 162616
3434 015272 005067 163234
3435 015276 012767 000001 163230
3436 015304 000207
3437
3438
3439
3440
3441
3442
3443 015306 016700 164176
3444 015312 001004
3445 015314 004767 002346

*****
* SUBROUTINE TO RELOCATE PROGRAM BACK TO BANKS 0 AND 1.
*****
RELO: BIT #1, PRGMAP ;CHECK FOR PROGRAM ALREADY IN BANKS 0 OR 1.
      BEQ 1$, ;BR IF NO CONFLICT.
      JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
;FATAL ERROR
      .WORD 7 ;ERROR TYPE CODE.
      HALT ;FATAL ERROR!!! PROGRAM ALREADY IN BANKS 0 OR 1!!!!
1$: TST MMAVA ;CHECK FOR MEM MGMT.
    BEQ 10$, ;BR IF NO MEMMGMT.
    CLR @#KIPAR2 ;SET PAR 2 TO BANK 0.
    MOV #200, @#KIPAR3 ;SET PAR 3 TO BANK 1.
    JSR R5, RELOC ;GO MOVE 8K INTO BANKS 0 AND 1.
      .WORD 0 ;SOURCE STARTING ADDRESS.
      .WORD 40000 ;DESTINATION STARTING ADDRESS.
    CLR @#KIPAR0 ;RESTORE PAR 0 TO BANK 0.
    MOV #200, @#KIPAR1 ;RESTORE PAR 1 TO BANK 1.
; * PROGRAM IS NOW EXECUTING OUT OF BANKS 0 AND 1.
    BR 30$, ;BR TO COMMON EXIT.

10$: MOV RELOCF, -(SP) ;PUT RELOCATION FACTOR ONTO THE STACK.
     MOV (SP), 20$, ;SET DATA FOR RELOC SUBROUTINE.
     JSR R5, RELOC ;GO MOVE THE PROGRAM BACK TO BANKS 0 AND 1.
20$: .WORD 0 ;SOURCE STARTING ADDRESS.
     .WORD 0 ;DESTINATION STARTING ADDRESS.
     SUB (SP), PC ;JUMP TO RELOCATED PROGRAM.
; * THE PROGRAM IS NOW EXECUTING OUT OF BANKS 0 AND 1.
     SUB (SP), SP ;RESET THE STACK POINTER.
     MOV R0, -(SP) ;PUSH R0 ON STACK
     MOV #RADTAB, R0 ;SET UP POINTER TO RELATIVE ADDRESS TABLE.

21$: SUB 2(SP), (R0)+ ;RESET ADDRESSES TO UNRELOCATED VALUES.
22$: TST (R0)+ ;CHECK FOR TERMINATORS.
     BEQ 22$, ;BR OVER TERMINATORS.
     CMP -(R0), #-1 ;CHECK FOR END OF TABLE INDICATOR.
     BNE 21$, ;BR IF MORE ADDRESSES IN TABLE.
     MOV (SP)+, R0 ;POP STACK INTO R0
     SUB (SP), @#ERRVEC ;ADJUST ERROR VECTOR.
     SUB (SP), @#PWRVEC ;ADJUST POWER FAIL VECTOR.
     SUB (SP), @#PARVEC ;ADJUST PARITY ERROR VECTOR.
     SUB (SP), SWR ;ADJUST SOFTWARE SWITCH REGISTER.
     SUB (SP), DISPLAY ;ADJUST SOFTWARE DISPLAY REGISTER.
23$: SUB (SP)+, (SP) ;ADJUST RETURN ADDRESS.
30$: CLR RELOCF ;RESET RELOCATION FACTOR.
     MOV #1, PRGMAP ;SET PROGRAM MAP TO POINT TO BANKS 0 AND 1.
     RTS PC ;RETURN.

*****
; * THIS SUBROUTINE MOVES THE LOADER AREA BACK TO THE 'TOP' OF MEMORY FROM
; * WHENCE IT CAME. THE LOADER AREA IS SAVED AT THE END OF THE 8K OF
; * PROGRAM CODE WHEN THE PROGRAM IS INITIALLY RUN.
*****
RESLDR: MOV LMAD, R0 ;CHECK IF THE LOADERS WERE SAVED.
        BNE 1$, ;BR IF LOADER AREA WAS SAVED.
        JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)

```

```

3446                                     ;FATAL ERROR
3447 015320 000007                       .WORD 7
3448 015322 000000                       HALT
3449 015324 005767 163206                1$: TST MAVA
3450 015330 001402                       BEQ 2$
3451 015332 005037 177572                CLR @#SRO
3452 015336 012701 040000                2$: MOV #40000, R1
3453 015342 012702 003000                MOV #1536., R2
3454 015346 014140                       3$: MOV -(R1), -(R0)
3455 015350 005302                       DEC R2
3456 015352 001375                       BNE 3$
3457 015354 005067 164130                CLR LMAD
3458 015360 005767 163152                TST MAVA
3459 015364 001402                       BEQ 4$
3460 015366 005237 177572                INC @#SRO
3461 015372 000207                       4$: RTS PC
3462
3463                                     ;* ROUTINE TO SAVE THE LOADERS AT THE END OF 8K.
3464 015374 005767 164110                SAVLDR: TST LMAD
3465 015400 001024                       BNE 4$
3466 015402 012700 040000                MOV #40000, R0
3467 015406 010001                       MOV R0, R1
3468 015410 012737 015422 000004        1$: MOV #2$, @#ERRVEC
3469 015416 011020                       MOV (R0), (R0)+
3470 015420 000776                       BR 1$
3471 015422 022626                       2$: CMP (SP)+, (SP)+
3472 015424 012737 023252 000004        MOV #ERRTRP, @#ERRVEC
3473 015432 010046                       MOV R0, -(SP)
3474 015434 012702 003000                MOV #1536., R2
3475 015440 014041                       3$: MOV -(R0), -(R1)
3476 015442 005302                       DEC R2
3477 015444 001375                       BNE 3$
3478 015446 012667 164036                MOV (SP)+, LMAD
3479 015452 000207                       4$: RTS PC

```

```

;ERROR TYPE CODE.
;FATAL ERROR!!! CAN'T RESTORE LOADER AREA IF IT WASN'T SAVED.
;CHECK FOR MEM MGMT.
;SKIP IF NO MEM MGMT.
;DISABLE MEM MGMT.
;GET END OF 8K, ASSUME PROG NOT RELOCATED.
;GET COUNTER.
;MOVE THE LOADER AREA.
;COUNT HOW LONG THE AREA IS.
;BR IF NOT MORE TO MOVE.
;CLEAR MONITOR SAVED FLAG
;CHECK FOR MEM MGMT.
;BR IF NO MEM MGMT.
;ENABLE MEM MGMT.
;RETURN.

;CHECK IF LOADERS HAVE BEEN SAVED ALREADY.
;BRANCH IF ALREADY SAVED
;GET END OF 8K
;GET END OF 8K
;SET UP TIMEOUT VECTOR
;SEARCH FOR END OF MEMORY
;KEEP SEARCHING
;RESTORE STACK POINTER
;RESET TIMEOUT VECTOR.
;SAVE LAST MEMORY ADDRESS (CONTIGUOUS)
;SET UP WORD COUNTER
;SAVE THE LOADERS
;COUNT THE WORDS
;BRANCH IF MORE WORDS
;SAVE LAST MEMORY ADDRESS
;RETURN

```


3480
3481
3482
3483
3484
3485
3486 015454 011667 163432
3487 015460 004567 004110
3488 015464 024465
3489
3490 015466 010146
3491 015470 010346
3492 015472 016703 164324
3493 015476 005733
3494 015500 100415
3495 015502 005713
3496 015504 001374
3497 015506 004767 002154
3498
3499 015512 000024
3500 015514 000417
3501 015516 005713
3502 015520 001415
3503 015522 005733
3504 015524 100374
3505 015526 004567 004042
3506 015532 024556
3507
3508 015534
3509 015534 004767 001010
3510 015540 004767 002122
3511 015544 000025
3512 015546 004767 000262
3513 015552 000761
3514 015554
3515 015554 012603
3516 015556 012601
3517 015560 000002
3518
3519
3520
3521
3522
3523 015562 005767 165602
3524 015566 001434
3525 015570 032777 000100 163332
3526 015576 001030
3527 015600 005767 162726
3528 015604 001410
3529 015606 032777 000040 163314
3530 015614 001004
3531 015616 026727 164140 001000
3532 015624 103415
3533
3534 015626 016737 164176 000114
3535 015634 005037 000116

```
.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.
*****
PESRV: MOV (SP), $BDADR ;GET PC OF INSTRUCTION WHICH CAUSED ERROR.
        JSR R5, $SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
        .WORD UNEXPT ;ADDRESS OF MESSAGE TO BE TYPED
        ;'UNEXPECTED MEMORY PARITY TRAP.'
        MOV R1, -(SP) ;PUSH R1 ON STACK
        MOV R3, -(SP) ;PUSH R3 ON STACK
        MOV .MPRX, R3 ;GET POINTER TO PARITY REGISTERS.
1$: TST @ (R3)+ ;CHECK THE PARITY REG FOR AN ERROR FLAG.
    BMI 3$ ;BR IF THIS REGISTER SHOWS THE ERROR.
    TST (R3) ;CHECK FOR TABLE TERMINATOR.
    BNE 1$ ;BR IF MORE REGISTERS.
    JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
;***ERROR*** NO REGISTER INDICATED ERROR
        .WORD 24 ;ERROR TYPE CODE.
        BR 4$ ;EXIT
2$: TST (R3) ;CHECK FOR TABLE TERMINATOR.
    BEQ 4$ ;BR IF NO MORE PARITY REGISTERS.
    TST @ (R3)+ ;CHECK THE PARITY REG FOR AN ERROR FLAG.
    BPL 2$ ;BR IF NO ERROR FLAG.
    JSR R5, $SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
    .WORD MTOE ;ADDRESS OF MESSAGE TO BE TYPED
    ;'MORE THAN ONE ERROR FOUND.'
3$: JSR PC, $SPRINTQ ;SET UP VALUES FOR ERROR PRINTING.
64$: JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
    .WORD 25 ;ERROR TYPE CODE.
    JSR PC, $PSCAN ;GO SCAN MEMORY FOR BAD PARITY.
    BR 2$ ;GO LOOK FOR MORE ERRORS.
4$: MOV (SP)+, R3 ;POP STACK INTO R3
    MOV (SP)+, R1 ;POP STACK INTO R1
    RTI ;RETURN.
*****
;ROUTINE TO ENABLE PARITY ERROR ACTION ON MA/MF PARITY MEMORIES
;THIS ROUTINE IS MEANT TO CATCH UNEXPECTEDS
*****
MAMF: TST MPRX ;CHECK IF ANY PARITY REGISTERS EXIST.
      BEQ MAMF2 ;EXIT IF NO PARITY REGISTERS.
      BIT #SW6, @SWR ;CHECK FOR INHIBIT PARITY ERROR DETECTION.
      BNE MAMF2 ;EXIT IF NO PARITY ERROR DETECTION.
      TST RELOCF ;CHECK IF PROGRAM RELOCATED OUT OF BANK 0.
      BEQ SETAE ;BR IF PROG IN BANK 0.
      BIT #SW5, @SWR ;CHECK IF VECTORS PROTECTED.
      BNE SETAE ;BR IF VECTOR AREA PROTECTED.
      CMP FSTADR, #1000 ;CHECK FOR STARTING ADDRESS ABOVE THE VECTORS.
      BLO MAMF2 ;EXIT IF VECTORS EXPOSED TO TESTING.
SETAE: MOV .PESRV, @#PARVEC ;SET PARITY ERROR TRAP VECTOR
        CLR @#PARVEC+2 ;PRIORITY LEVEL 0 ON TRAP
```

3536 015640 010346
3537 015642 016703 164154
3538 015646 052733 000001
3539 015652 005713
3540 015654 001374
3541 015656 012603
3542 015660 000207
3543
3544
3545
3546
3547 015662
3548 015662 005767 165502
3549 015666 001437
3550 015670 032777 000100 163232
3551 015676 001033
3552 015700 010346
3553 015702 016703 164114
3554 015706 005733
3555 015710 100023
3556 015712 032773 000001 177776
3557 015720 001410
3558 015722 004767 000622
3559 015726 004767 001734
3560 015732 000026
3561 015734 000411
3562 015736 004767 000072
3563 015742
3564 015742 004767 000602
3565 015746 004767 001714
3566 015752 000027
3567 015754 004767 000054
3568 015760 005713
3569 015762 001351
3570 015764 012603
3571 015766 000207
3572
3573
3574
3575
3576
3577
3578
3579
3580 015770
3581 015770 010046
3582 015772 010146
3583 015774 010446
3584 015776 012700 000020
3585 016002 016701 164006
3586 016006 010304
3587 016010 005724
3588 016012 032124
3589 016014 001003
3590 016016 005300
3591 016020 001374

```
MOV R3,-(SP) ;:PUSH R3 ON STACK
MOV .MPRX, R3 ;:GET PARITY REGISTER TABLE POINTER.
MAMF1: BIS #AE, @(R3)+ ;:SET ACTION ENABLE BIT IN PARITY REG
TST (R3) ;:CHECK FOR END OF TABLE.
BNE MAMF1 ;:BR IF MORE PARITY REGISTERS.
MOV (SP)+,R3 ;:POP STACK INTO R3
MAMF2: RTS PC ;:RETURN.

;:*****
;:* SUBROUTINE TO CHECK PARITY REGISTERS FOR ERRORS THAT DIDN'T TRAP.
;:*****
CHGG2:
CKPMER: TST MPRX ;:CHECK IF ANY PARITY REGISTERS EXIST.
BEQ 4$ ;:BR IF NO PARITY REGISTERS.
BIT #SW6, @SWR ;:CHECK FOR INHIBIT PARITY ERROR CHECKING.
BNE 4$ ;:BR IF PARITY ERROR CHECKING INHIBITED.
MOV R3,-(SP) ;:PUSH R3 ON STACK
MOV .MPRX, R3 ;:GET PARITY REG TABLE POINTER.
1$: TST @(R3)+ ;:CHECK THE PARITY REG FOR AN ERROR FLAG.
BFL 3$ ;:BR IF NO ERROR
BIT #BIT0, @-2(R3) ;:CHECK IF A TRAP SHOULD HAVE OCCURRED.
BEQ 2$ ;:BR IF NO ACTION ENABLE. CHGG2
64$: JSR PC, SPRNTQ ;:SET UP VALUES FOR ERROR PRINTING.
JSR PC, $ERROR ;:*** ERROR *** (GO TYPE A MESSAGE)
WORD 26 ;:ERROR TYPE CODE.
BR 3$
JSR PC, PSCAN ;:GO SCAN ALL MEMORY FOR PARITY ERRORS.
2$:
65$: JSR PC, SPRNTQ ;:SET UP VALUES FOR ERROR PRINTING.
JSR PC, $ERROR ;:*** ERROR *** (GO TYPE A MESSAGE)
WORD 27 ;:ERROR TYPE CODE.
JSR PC, PSCAN ;:GO SCAN ALL MEMORY FOR PARITY ERRORS.
3$: TST (R3) ;:CHECK FOR TABLE TERMINATOR.
BNE 1$ ;:BR IF MORE.
MOV (SP)+,R3 ;:POP STACK INTO R3
4$: RTS PC ;:RETURN.

;:*****
;:* SUBROUTINE TO VERIFY THAT THE PARITY CSR DOES CONTROL THE BANK
;:* UNDER TEST. MATCH THE BANK UNDER TEST WITH THE PARITY MAP TABLE.
;:* RETURN IF MEM BANK HAS BEEN MAPPED IN PARITY MAP TABLE.
;:* RETURN+2 IF ERROR, MEM BANK HAS NOT BEEN MAPPED.
;:*****
PARMAT:
MOV R0,-(SP) ;:PUSH R0 ON STACK
MOV R1,-(SP) ;:PUSH R1 ON STACK
MOV R4,-(SP) ;:PUSH R4 ON STACK
MOV #20,R0 ;:COUNT OF PARITY MAP TABLE
MOV .BITPT,R1 ;:MEM BANK UNDER TEST POINTER
MOV R3,R4 ;:STORE PARITY MAP TABLE
TST (R4)+ ;:MOVE PAST CSR ADDRESS TO MAP TABLE
1$: BIT (R1)+,(R4)+ ;:CHECK FOR TABLE MATCH
BNE 2$ ;:BR IF MATCH MADE IN TABLES
DEC R0 ;:DO NEXT WORD IN TABLE
BNE 1$ ;:BR TO NEXT WORD IN TABLE
```


3592 016022 005725
3593 016024
3594 016024 012604
3595 016026 012601
3596 016030 012600
3597 016032 000205
3598
3599
3600
3601
3602
3603
3604
3605
3606
3607
3608
3609
3610
3611
3612
3613 016034
3614 016034 010046
3615 016036 010146
3616 016040 010246
3617 016042 010346
3618 016044 010446
3619 016046 004567 003522
3620 016052 024622
3621
3622 016054 004767 000434
3623 016060 013746 000114
3624 016064 013746 000116
3625 016070 012737 000116 000114
3626 016076 005037 000116
3627 016102 005004
3628 016104 005002
3629 016106 012700 000001
3630 016112 016701 163674
3631 016116 005767 162414
3632 016122 001413
3633 016124 013746 172344
3634 016130 013746 172346
3635 016134 005037 172344
3636 016140 012737 000200 172346
3637 016146 012702 040000
3638
3639 016152 030011
3640 016154 001003
3641 016156 062702 040000
3642 016162 000475
3643
3644 016164 111267 163600
3645 016170 016703 163626
3646
3647 016174 005733

```

TST      (R5)+          ;NO MATCH MADE,RETURN +2
2$:      MOV      (SP)+,R4      ;;POP STACK INTO R4
        MOV      (SP)+,R1      ;;POP STACK INTO R1
        MOV      (SP)+,R0      ;;POP STACK INTO R0
        RTS      R5
    
```

```

*****
* AFTER A PARITY ERROR IS ENCOUNTERED THIS SUBROUTINE IS USED
* TO SCAN ALL PARITY MEMORY PER THE PMEMAP TABLE LOOKING FOR BYTE LOCATIONS
* CAUSING THE PARITY ERROR. UPON FINDING THE LOCATION(S) A MESSAGE
* WILL BE TYPED AND THE LOCATION WILL BE REWRITTEN IN ORDER TO RESTORE GOOD PARITY.
* STORAGE USED:
*   R0      = BANK POINTER
*   R1      = PARITY CSR TABLE POINTER (MPRX)
*   R2      = ADDRESS POINTER
*   R3      = PMEMAP TABLE ENTRY
*   R4      = ERROR DETECT FLAG
*   TEMP    = TEMPORARY STORAGE
*****
    
```

```

PSCAN:
MOV      R0,-(SP)      ;;PUSH R0 ON STACK
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
JSR      R5,$SPRINT    ;GO PRINT OUT THE FOLLOWING MESSAGE.
        .WORD    SCANM    ;ADDRESS OF MESSAGE TO BE TYPED
        ;'SCANNING MEMORY FOR BAD PARITY'
JSR      PC,CLRPAR     ;GO INITIALIZE PARITY CSR'S
MOV      @#114,-(SP)   ;;PUSH @#114 ON STACK
MOV      @#116,-(SP)   ;;PUSH @#116 ON STACK
MOV      #116,@#114    ;SET UP PARITY VECTOR TO
CLR      @#116        ;HALT IF ANOTHER PARITY ERROR OCCURS
CLR      R4           ;INITIALIZE ERROR DETECT FLAG
CLR      R2           ;INITIALIZE ADDRESS POINTER
MOV      #BIT0,R0     ;INITIALIZE BANK POINTER
MOV      .PMEMAP,R1   ;POINT TO PMEMAP TABLE
TST      MMAVA        ;IF MEMORY MANAGEMENT NOT AVAILABLE
BEQ      1$          ; THEN BRANCH
MOV      @#KIPAR2,-(SP) ;;PUSH @#KIPAR2 ON STACK
MOV      @#KIPAR3,-(SP) ;;PUSH @#KIPAR3 ON STACK
CLR      @#KIPAR2     ;INITIALIZE PAR2 TO 0
MOV      #200,@#KIPAR3 ;INITIALIZE PAR3
MOV      #40000,R2    ;INITIALIZE ADDRESS POINTER TO MAP THRU PAR2
1$:      BIT      R0,(R1)      ;IF 8K BANK DOES EXIST
        BNE      2$          ; THEN BRANCH
        ADD      #40000,R2    ; ELSE UPDATE ADDRESS TO NEXT 8K BANK
        BR      10$         ; AND BRANCH TO SEE IF SHOULD CONTINUE TESTING
2$:      MOVB     (R2),TEMP    ;READ THE BYTE LOCATION
        MOV      .MPRX,R3    ;POINT TO PARITY CSR TABLE
3$:      TST      @ (R3)+      ;IF NO ERROR FLAG SET IN CSR
    
```

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


```

3704 016402 012700 000001      MOV    #BIT0,R0      ; ELSE REINIT BANK POINTER
3705
3706 016406 005767 162124      11$:  TST    MAVA      ;IF KT AVAILABLE
3707 016412 001004                BNE    12$          ; THEN BRANCH
3708 016414 032700 000020      BIT    #BIT4,R0     ; ELSE IF AT 32K LIMIT
3709 016420 001016                BNE    14$          ; THEN ALL DONE CHECKING
3710 016422 000653                BR     1$           ; ELSE TRY SOMEMORE
3711
3712 016424 062737 000400 172344 12$:  ADD    #400,@#KIPAR2 ;UPDATE PARS TO MAP TO NEXT
3713 016432 062737 000400 172346      ADD    #400,@#KIPAR3 ; 8K BANK
3714 016440 012702 040000      MOV    #40000,R2    ;REINIT ADDRESS
3715 016444 000642                BR     1$           ;AND TRY SOMEMORE
3716
3717 016446                13$:
3718 016446 012637 172346      MOV    (SP)+,@#KIPAR3 ;:POP STACK INTO @#KIPAR3
3719 016452 012637 172344      MOV    (SP)+,@#KIPAR2 ;:POP STACK INTO @#KIPAR2
3720
3721 016456 005704                14$:  TST    R4           ;IF ERROR FOUND
3722 016460 001003                BNE    15$          ; THEN BRANCH
3723 016462 004567 003106      JSR    R5,$SPRINT   ;GO PRINT OUT THE FOLLOWING MESSAGE.
3724 016466 024026                .WORD  NOPE$       ;ADDRESS OF MESSAGE TO BE TYPED
3725
3726 016470                15$:
3727 016470 012637 000116      MOV    (SP)+,@#116  ;:POP STACK INTO @#116
3728 016474 012637 000114      MOV    (SP)+,@#114  ;:POP STACK INTO @#114
3729 016500 012604                MOV    (SP)+,R4     ;:POP STACK INTO R4
3730 016502 012603                MOV    (SP)+,R3     ;:POP STACK INTO R3
3731 016504 012602                MOV    (SP)+,R2     ;:POP STACK INTO R2
3732 016506 012601                MOV    (SP)+,R1     ;:POP STACK INTO R1
3733 016510 012600                MOV    (SP)+,R0     ;:POP STACK INTO R0
3734
3735 016512 000207                16$:  RTS     PC          ;RETURN
3736
3737
3738
3739
3740

```

 :ROUTINE TO CLEAR ALL PARITY REGISTERS PRESENT

```

3741 016514                CLRPAR:
3742 016514 010346                MOV    R3,-(SP)     ;:PUSH R3 ON STACK
3743 016516 016703 163300      MOV    .MPRX,R3     ;GET PARITY REGISTER TABLE POINTER.
3744 016522 005713                1$:  TST    (R3)         ;CHECK FOR THE TABLE TERMINATOR.
3745 016524 001402                BEQ    2$           ;BR IF DONE ALL PARITY REGISTERS.
3746 016526 005033                CLR    @(R3)+       ;CLEAR THE PARITY REGISTER.
3747 016530 000774                BR     1$           ;BR FOR MORE
3748 016532                2$:
3749 016532 012603                MOV    (SP)+,R3     ;:POP STACK INTO R3
3750 016534 000207                RTS     PC          ;RETURN.
3751
3752
3753
3754
3755
3756
3757

```

.SBTTL SUBROUTINES TO SET UP DATA FOR ERROR PRINTOUT ROUTINE.

 :* THESE ROUTINES ARE USED TO TRANSFER DATA TO COMMON TAG AREA (.SCMTAG)
 :* FOR ERROR PRINTOUT BY .\$ERROR & .\$ERRTYP ROUTINES FROM **SYSMAC**.

```

3758 016536 010267 162346      $PRNT: MOV    R2,$GDADR ;SAVE THE ADDRESS UNDER TEST.
3759 016542 005067 162346      CLR    $GDDAT      ;SHOULD BE DATA IS '0'.

```

```

3760 016546 000435          BR      SPRNTB
3761
3762 016550 014367 162374  SPRNTQ: MOV    -(R3), $TMP0    ;GET THE PARITY REGISTER ADDRESS.
3763 016554 013367 162372          MOV    @R3+, $TMP1    ;GET THE CONTENTS OF THE PARITY REG.
3764 016560 016767 163204 162330  MOV    TEMP,$BDDAT
3765 016566 010267 162316          MOV    R2,$GDADR
3766 016572 000425          BR      SPREND
3767
3768 016574 011367 162350  SPRNTP: MOV    (R3), $TMP0    ;GET THE PARITY REGISTER ADDRESS.
3769 016600 010267 162304  SPRNT0: MOV    R2, $GDADR    ;GET THE MEMORY ADDRESS BEING TESTED
3770 016604 000414          BR      SPRNTA    ;BR TO COMMON SECTION.
3771
3772 016606 010267 162276  SPRNT1: MOV    R2, $GDADR    ;GET THE MEMORY ADDRESS BEING TESTED
3773 016612 005367 162272          DEC    $GDADR    ;ADJUST IT FOR PRINTOUT.
3774 016616 000407          BR      SPRNTA    ;BR TO COMMON SECTION.
3775
3776 016620 010367 162324  SPRNT3: MOV    R3, $TMP0    ;GET THE DATA IN R3.
3777 016624 010267 162260  SPRNT2: MOV    R2, $GDADR    ;GET THE MEMORY ADDRESS BEING TESTED
3778 016630 162767 000002 162252  SUB    #2, $GDADR    ;ADJUST IT FOR PRINTOUT.
3779 016636 010067 162252  SPRNTA: MOV    R0, $GDADR    ;GET WHAT THE DATA SHOULD BE
3780 016642 010167 162250  SPRNTB: MOV    R1, $BDDAT    ;GET WHAT THE DATA WAS
3781 016646 000207  SPREND: RTS    PC    ;RETURN TO ENTER ERROR ROUTINES
3782
3783
3784
3785
3786
3787
3788
3789
3790
3791
3792
3793
3794
3795
3796
3797
3798
3799
3800

```

```

*****
* 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 = CONSECUTIVE 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
*****

```

```

3801 016650 010037 017350  TYPMAP: MOV    R0,@#BNKTAB    ;SAVE ADDRESS OF TABLE
3802 016654 012737 000040 162352  MOV    #40,@#ENDBKT    ;LOAD END BANK TABLE WITH 40
3803 016662 060037 017352          ADD    R0,@#ENDBKT    ;GET LAST ADR + 2 INTO ENDBKT
3804 016666 005710 1$:      TST    (R0)    ;ANY MEMORY IN THIS TABLE ENTRY
3805 016670 001011          BNE    2$    ; THEN GO PRINT OUT LOCATIONS
3806 016672 005720          TST    (R0)+    ; ELSE POINT TO NEXT TABLE ENTRY
3807 016674 023700 017352          CMP    @#ENDBKT,R0    ;IF NOT END OF TABLE
3808 016700 001372          BNE    1$    ; THEN TRY AGAIN
3809 016702 004567 002666          JSR    R5, $PRINT    ;GO PRINT OUT THE FOLLOWING MESSAGE.
3810 016706 024252          .WORD  NOMEM    ;ADDRESS OF MESSAGE TO BE TYPED
3811          ;'NO MEMORY FOUND.'
3812 016710 000167 000426          JMP    13$    ;EXIT
3813
3814 016714 013700 017350  2$:      MOV    @#BNKTAB,R0    ;RESTORE MAP TABLE ADDRESS
3815 016720 010146          MOV    R1,-(SP)    ;PUSH R1 ON STACK

```



```

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

```



```

3928
3929 017306 005720
3930 017310 023700 017352
3931 017314 001220
3932 017316 005760 177776
3933 017322 100003
3934 017324 005267 162440
3935 017330 000623
3936 017332
3937 017332 012604
3938 017334 012603
3939 017336 012602
3940 017340 012601
3941
3942 017342 000207
3943
3944 017344 000000
3945 017346 000000
3946 017350 000000
3947 017352 000000
3948
3949
3950
3951
3952
3953
3954
3955
3956
3957
3958
3959
3960
3961
3962
3963
3964 017354
3965
3966
3967 017354 106746
3968 017356 105066 000001
3969
3970 017362 004767 001042
3971 017366 010516
3972 017370 032777 040000 161532
3973 017376 001117
3974
3975 017400 000416
3976
3977 017402 013746 000004
3978 017406 012737 017426 000004
3979 017414 005737 177060
3980 017420 012637 000004
3981 017424 000466
3982 017426 022626
3983 017430 012637 000004

12$: TST (R0)+ ;UPDATE POINTER TO NEXT TABLE ENTRY
      CMP @#ENDBKT,R0 ;IF NOT DONE WITH TABLE
      BNE 3$ ; THEN TRY AGAIN
      TST -2(R0) ;CHECK IF LAST BANK AVAILABLE
      BPL 14$ ;IF ZERO, TYPMAP PRINT COMPLETED
      INC TEMP ;ELSE SET LAST BANK PRINT FLAG
      BR 5$ ;PRINT FOR LAST BANK

14$: MOV (SP)+,R4 ;:POP STACK INTO R4
      MOV (SP)+,R3 ;:POP STACK INTO R3
      MOV (SP)+,R2 ;:POP STACK INTO R2
      MOV (SP)+,R1 ;:POP STACK INTO R1

13$: RTS PC ;RETURN TO CALLER

LMEMLO: .WORD 0 ; LOW MEMORY ADR ... LO 16 BITS
LMEMHI: .WORD 0 ; LOW MEMORY ADR ... HI 6 BITS
BNKTAB: .WORD 0 ; ADDRESS OF TABLE TO BE PROCESSED
ENDBKT: .WORD 0 ; LAST ADDRESS + 2 OF TABLE
; TO BE PROCESSED.

.SBTTL SCOPE HANDLER ROUTINE

;*****
;THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
;AND LOAD THE TEST NUMBER($STNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
;AND LOAD THE ERROR FLAG ($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

$SCOPE:
;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $CKSWR ROUTINE
;* WIHTOUT USING A 'TRAP' INSTRUCTION AS CALLED FOR BY **SYSMAC**.
MFPS -(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
CLRB 1(SP) ;HIGH BYTE CLEARED TO INSURE KERNEL MODE
;ON PSW RETURN.
JSR PC, $CKSWR ;GO TO THE SUBROUTINE
MOV R5, (SP) ;PUT RETURN PC ONTO STACK, SIMULATE JSR PC.
1$: BIT #BIT14,@SWR ;:LOOP ON PRESENT TEST?
      BNE $OVER ;:YES IF SW14=1
;#####START OF CODE FOR THE XOR TESTER#####
$XTSTR: BR 6$ ;:IF RUNNING ON THE 'XOR' TESTER CHANGE
;THIS INSTRUCTION TO A 'NOP' (NOP=240)
MOV @#ERRVEC,-(SP) ;:SAVE THE CONTENTS OF THE ERROR VECTOR
MOV #5$,@#ERRVEC ;:SET FOR TIMEOUT
TST @#177060 ;:TIME OUT ON XOR?
MOV (SP)+,@#ERRVEC ;:RESTORE THE ERROR VECTOR
BR $SVLAD ;:GO TO THE NEXT TEST
5$: CMP (SP)+,(SP)+ ;:CLEAR THE STACK AFIER A TIME OUT
      MOV (SP)+,@#ERRVEC ;:RESTORE THE ERROR VECTOR

```

```

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


4040
4041
4042 017666
4043
4044
4045 017666 106746
4046 017670 105066 000001
4047
4048 017674 004767 000530
4049 017700 062716 000002
4050 017704 105267 161163
4051 017710 001775
4052 017712 016777 161154 161212
4053 017720 032777 002000 161202
4054 017726 001403
4055 017730 004567 001640
4056 017734 001164
4057 017736 005267 161140
4058 017742 011667 161140
4059 017746 162767 000002 161132
4060 017754 117767 161126 161122
4061 017762 032777 020000 161140
4062 017770 001005
4063 017772 004767 000120
4064 017776 004567 001572
4065 020002 001171
4066 020004
4067 020004 122767 000001 161202
4068 020012 001007
4069 020014 116767 161064 000004
4070 020022 004767 002152
4071 020026 000
4072 020027 000
4073 020030 000777
4074 020032 005777 161072
4075 020036 100006
4076 020040 000000
4077
4078
4079 020042 106746
4080 020044 105066 000001
4081
4082 020050 004767 000354
4083 020054 032777 001000 161046
4084 020062 001402
4085 020064 016716 161010
4086 020070 005767 161066
4087 020074 001402
4088 020076 016716 161060
4089 020102
4090 020102 022737 012560 000042
4091 020110 001001
4092 020112 000000
4093 020114
4094 020114 000207
4095

```
;* ERROR N ;;ERROR=EMT AND N=ERROR ITEM NUMBER

$ERROR:
;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $CKSWR ROUTINE
;* WIHTOUT USING A 'TRAP' INSTRUCTION AS CALLED FOR BY **SYSMAC**.
MFPS -(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
CLRB 1(SP) ;HIGH BYTE CLEARED TO INSURE KERNEL MODE
;ON PSW RETURN.
JSR PC, $CKSWR ;GO TO THE SUBROUTINE
ADD #2, (SP) ;ADJUST POINTER PAST CODE WORD.
7$: INCB $ERFLG ;SET THE ERROR FLAG
BEQ 7$ ;DON'T LET THE FLAG GO TO ZERO
MOV $TSTNM,@DISPLAY ;DISPLAY TEST NUMBER AND ERROR FLAG
BIT #BIT10,@SWR ;BELL ON ERROR?
BEQ 1$ ;NO - SKIP
JSR R5, $SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
;ADDRESS OF MESSAGE TO BE TYPED
1$: INC $ERTTL ;COUNT THE NUMBER OF ERRORS
MOV (SP), $ERRPC ;GET ADDRESS OF ERROR INSTRUCTION
SUB #2, $ERRPC
MOVB @ $ERRPC, $ITEMB ;;STRIP AND SAVE THE ERROR ITEM CODE
BIT #BIT13,@SWR ;SKIP TYPEOUT IF SET
BNE 20$ ;SKIP TYPEOUTS
JSR PC, $ERRTYP ;GO TO USER ERROR ROUTINE
JSR R5, $SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
;ADDRESS OF MESSAGE TO BE TYPED
20$: CMPB #APTENV, $ENV ;;RUNNING IN APT MODE
BNE 2$ ;NO, SKIP APT ERROR REPORT
MOVB $ITEMB, 21$ ;SET ITEM NUMBER AS ERROR NUMBER
JSR PC, $ATY4 ;REPORT FATAL ERROR TO APT
21$: .BYTE 0
;BYTE 0
22$: BR 22$ ;;APT ERROR LOOP
2$: TST @SWR ;HALT ON ERROR
BPL 3$ ;SKIP IF CONTINUE
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**.
MFPS -(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
CLRB 1(SP) ;HIGH BYTE CLEARED TO INSURE KERNEL MODE
;ON PSW RETURN.
JSR PC, $CKSWR ;GO TO THE SUBROUTINE
3$: BIT #BIT09,@SWR ;LOOP ON ERROR SWITCH SET?
BEQ 4$ ;BR IF NO
MOV $LPERR, (SP) ;FUDGE RETURN FOR LOOPING
4$: TST $ESCAPE ;CHECK FOR AN ESCAPE ADDRESS
BEQ 5$ ;BR IF NONE
MOV $ESCAPE, (SP) ;FUDGE RETURN ADDRESS FOR ESCAPE
5$: CMP # $ENDAD, @#42 ;ACT-11 AUTO-ACCEPT?
BNE 6$ ;BRANCH IF NO
HALT ;YES
6$: RTS PC
;*****
```

4096
 4097
 4098
 4099
 4100
 4101
 4102
 4103
 4104
 4105
 4106
 4107
 4108
 4109
 4110
 4111
 4112
 4113
 4114
 4115
 4116
 4117
 4118
 4119
 4120
 4121
 4122
 4123
 4124
 4125
 4126
 4127
 4128
 4129
 4130
 4131
 4132
 4133
 4134
 4135
 4136
 4137
 4138
 4139
 4140
 4141
 4142
 4143
 4144
 4145
 4146
 4147
 4148
 4149
 4150
 4151

020116
 020116 004567 001452
 020122 001171
 020124 010046
 020126 005000
 020130 156700 160750
 020134 001010
 020136 016746 160744
 020142 106746
 020144 105066 000001
 020150 004767 002672
 020154 000516
 020156 016767 160724 161320 1\$:
 020164 166767 160342 161312
 020172 005300
 020174 006300
 020176 006300
 020200 006300
 020202 066700 161624
 020206 012067 000006
 020212 001406
 020214 004567 001354
 020220 000000 2\$:
 020222 004567 001346
 020226 001171
 020230 012067 000006 3\$:
 020234 001406
 020236 004567 001332 4\$:
 020242 000000
 020244 004567 001324
 020250 001171
 020252 010146 5\$:
 020254 012001
 020256 001454
 020260 066701 160246
 020264 012000
 020266 066700 160240
 020272 105720 6\$:
 020274 001007
 020276 013146

.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.

```

$ERRTYP:
      JSR      R5,      $SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
      .WORD   $CRLF      ;ADDRESS OF MESSAGE TO BE TYPED
      MOV     R0,-(SP)      ;SAVE R0
      CLR     R0           ;PICKUP THE ITEM INDEX
      BISB   $ITEMB,R0
      BNE    1$           ;IF ITEM NUMBER IS ZERO, JUST
                        ;TYPE THE PC OF THE ERROR
      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**.
      MFPS   -(SP)      ;PUT THE PROCESSOR STATUS ON THE STACK
      CLRB   1(SP)      ;HIGH BYTE CLEARED TO INSURE KERNEL MODE
                        ;ON PSW RETURN.
      JSR     PC,      $TYPOC ;GO TO THE SUBROUTINE
      BR     10$         ;GET OUT
      MOV     $ERRPC, $VERPC ;SET UP VIRTUAL PC FOR TYPEOUT.
      SUB    RELOCF, $VERPC ;MAKE VIRTUAL IF NOT ALREADY.
                        ;ADJUST THE INDEX SO THAT IT WILL
                        ;WORK FOR THE ERROR TABLE
      DEC    R0
      ASL    R0
      ASL    R0
      ASL    R0
      ADD    .ERRTB, R0 ;FORM TABLE POINTER
      MOV    (R0)+,2$   ;PICKUP "ERROR MESSAGE" POINTER
      BEQ    3$         ;SKIP TYPEOUT IF NO POINTER
      JSR    R5,      $SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
      .WORD   0         ;"ERROR MESSAGE" POINTER GOES HERE
      JSR    R5,      $SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
      .WORD   $CRLF      ;ADDRESS OF MESSAGE TO BE TYPED
      MOV    (R0)+,4$   ;PICKUP "DATA HEADER" POINTER
      BEQ    5$         ;SKIP TYPEOUT IF 0
      JSR    R5,      $SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
      .WORD   0         ;"DATA HEADER" POINTER GOES HERE
      JSR    R5,      $SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
      .WORD   $CRLF      ;ADDRESS OF MESSAGE TO BE TYPED
      MOV    R1,-(SP)   ;SAVE R1
      MOV    (R0)+,R1   ;PICKUP "DATA TABLE" POINTER
      BEQ    9$         ;BR IF NO DATA TO BE TYPED
      ADD    RELOCF, R1 ;ADJUST POINTER
      MOV    (R0)+,R0   ;PICKUP "DATA FORMAT" POINTER
      ADD    RELOCF, R0 ;ADJUST POINTER.
      TSTB   (R0)+     ;CHECK THE FORMAT
      BNE    7$         ;BR IF NOT 16-BIT OCTAL
      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**.
  
```



```

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

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


```

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

```

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


```

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

```

4432 021510 006102          ROL      R2
4433 021512 006301          ASL      R1          ;;*8
4434 021514 006102          ROL      R2
4435 021516 042716 177770  BIC      #^C7,(SP)   ;;STRIP THE ASCII JUNK
4436 021522 062601          ADD      (SP)+,R1    ;;ADD IN THIS DIGIT
4437 021524 000756          BR       2$          ;;LOOP
4438 021526 005726          3$: TST      (SP)+    ;;CLEAN TERMINATOR FROM STACK
4439 021530 010166 000012  MOV      R1,12(SP)   ;;SAVE THE RESULT
4440 021534 010267 000032  MOV      R2,$HIOCT
4441 021540 012602          MOV      (SP)+,R2    ;;POP STACK INTO R2
4442 021542 012601          MOV      (SP)+,R1    ;;POP STACK INTO R1
4443 021544 012600          MOV      (SP)+,R0    ;;POP STACK INTO R0
4444 021546 000002          RTI
4445 021550 005726          4$: TST      (SP)+    ;;CLEAN PARTIAL FROM STACK
4446 021552 105010          CLR      (R0)        ;;SET A TERMINATOR
4447 021554 004567 000014  JSR      R5, $SPRINT ;;GO PRINT OUT THE FOLLOWING MESSAGE.
4448 021560 000000          5$: .WORD    0
4449 021562 004567 000006  JSR      R5, $SPRINT ;;GO PRINT OUT THE FOLLOWING MESSAGE.
4450 021566 001170          .WORD    $QUES       ;;ADDRESS OF MESSAGE TO BE TYPED
4451 021570 000722          BR       1$          ;;TRY AGAIN
4452 021572 000000  $HIOCT: .WORD    0    ;;HIGH ORDER BITS GO HERE
4453
4454  ;;*****
4455  ;;* SUBROUTINE TO PASS RELOCATED MESSAGE ADDRESSES TO THE $TYPE ROUTINE.
4456  ;;* CALL:      JSR      R5, $SPRINT
4457  ;;*           <MESSAGE VIRTUAL ADDRESS>
4458  ;;*****
4459 021574 012567 000020  $SPRINT: MOV      (R5)+, 1$   ;;GET THE MESSAGE VIRTUAL ADDRESS.
4460 021600 066767 156726 000012  ADD      RELOC, 1$      ;;MAKE IT PHYSICAL.
4461  ;;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPE ROUTINE
4462  ;;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
4463 021606 106746          MFPS     -(SP)        ;;PUT THE PROCESSOR STATUS ON THE STACK
4464 021610 105066 000001  CLR      1(SP)        ;;HIGH BYTE CLEARED TO INSURE KERNEL MODE
4465  ;;* ON PSW RETURN.
4466 021614 004767 000004  JSR      PC, $TYPE     ;;GO TO THE SUBROUTINE
4467 021620 000000          .WORD    0            ;;CONTAINS THE PHYSICAL MESSAGE ADDRESS.
4468 021622 000205          RTS      R5          ;;RETURN.
4469
4470  .SBTTL  TYPE ROUTINE
4471
4472  ;;*****
4473  ;;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
4474  ;;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
4475  ;;*NOTE1:      $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
4476  ;;*NOTE2:      $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
4477  ;;*NOTE3:      $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
4478  ;;*
4479  ;;*CALL:
4480  ;;*1) USING A TRAP INSTRUCTION
4481  ;;*   TYPE      ,MESADR      ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
4482  ;;*OR
4483  ;;*   TYPE
4484  ;;*   MESADR
4485  ;;*
4486
4487 021624 105767 157317  $TYPE: TSTB   $TPFLG   ;;IS THERE A TERMINAL?
    
```



```

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

:MJD001
 :MJD001
 :MJD001
 :MJD001
 :MJD001
 :MJD001
 :MJD001

```

4544 022064 105777 157044      TSTB  @STKS      ;;WAIT FOR CHAR      :MJD001
4545 022070 100375      BPL    101$      :MJD001
4546 022072 117716 157040      MOVB  @STKB,(SP) ;;GET CHAR           :MJD001
4547 022076 042716 177600      BIC   #177600,(SP) ;;STRIP IT          :MJD001
4548 022102 122716 000021      CMPB  #$XON,(SP) ;;WAS IT XON?      :MJD001
4549 022106 001366      BNE   101$      ;;BR IF NOT        :MJD001
4550 022110      102$:      TST    (SP)+    ;;FIX STACK        :MJD001
4551 022110 005726      10$:      TST    (SP)+    :MJD001
4552 022112      TSTB  @STPS      ;;WAIT UNTIL PRINTER IS READY :MJD001
4553 022112 105777 157022      BPL    10$      :MJD001
4554 022116 100375      MOVB  2(SP),@STPB ;;LOAD CHAR TO BE TYPED INTO DATA REG.
4555 022120 116677 000002 157014      CMPB  #CR,2(SP) ;;IS CHARACTER A CARRIAGE RETURN?
4556 022126 122766 000015 000002      BNE   1$        ;;BRANCH IF NO
4557 022134 001003      CLRB  $CHARCNT  ;;YES--CLEAR CHARACTER COUNT
4558 022136 105067 000014      BR    $TYPEX    ;;EXIT
4559 022142 000406      CMPB  #LF,2(SP) ;;IS CHARACTER A LINE FEED?
4560 022144 122766 000012 000002 1$:      BEQ   $TYPEX    ;;BRANCH IF YES
4561 022152 001402      INCB  (PC)+    ;;COUNT THE CHARACTER
4562 022154 105227      $CHARCNT: .WORD 0 ;;CHARACTER COUNT STORAGE
4563 022156 000000      $TYPEX: RTS    PC
4564 022160 000207
4565
4566      .SBTTL  APT COMMUNICATIONS ROUTINE
4567
4568      ;*****
4569 022162 112767 000001 000376 $ATY1: MOV  #1,$FFLG ;;TO REPORT FATAL ERROR
4570 022170 112767 000001 000366 $ATY3: MOV  #1,$MFLG ;;TO TYPE A MESSAGE
4571 022176 000403      BR    $ATYC
4572 022200 112767 000001 000360 $ATY4: MOV  #1,$FFLG ;;TO ONLY REPORT FATAL ERROR
4573 022206      $ATYC:
4574 022206 010046      MOV   R0,-(SP) ;;PUSH R0 ON STACK
4575 022210 010146      MOV   R1,-(SP) ;;PUSH R1 ON STACK
4576 022212 105767 000346      TSTB  $MFLG    ;;SHOULD TYPE A MESSAGE?
4577 022216 001450      BEQ   5$      ;;IF NOT: BR
4578 022220 122767 000001 156766      CMPB  #APTENV,$ENV ;;OPERATING UNDER APT?
4579 022226 001031      BNE   3$      ;;IF NOT: BR
4580 022230 132767 000100 156757      BITB  #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
4581 022236 001425      BEQ   3$      ;;IF NOT: BR
4582 022240 017600 000004      MOV   @4(SP),R0 ;;GET MESSAGE ADDR.
4583 022244 062766 000002 000004      ADD   #2,4(SP) ;;BUMP RETURN ADDR.
4584 022252 005767 156716      1$:      TST   $MSGTYPE ;;SEE IF DONE W/ LAST XMISSION?
4585 022256 001375      BNE   1$      ;;IF NOT: WAIT
4586 022260 010067 156724      MOV   R0,$MSGAD ;;PUT ADDR IN MAILBOX
4587 022264 105720      2$:      TSTB  (R0)+    ;;FIND END OF MESSAGE
4588 022266 001376      BNE   2$
4589 022270 166700 156714      SUB   $MSGAD,R0 ;;SUB START OF MESSAGE
4590 022274 006200      ASR   R0      ;;GET MESSAGE LNGTH IN WORDS
4591 022276 010067 156710      MOV   R0,$MSGGLT ;;PUT LENGTH IN MAILBOX
4592 022302 012767 000004 156664      MOV   #4,$MSGTYPE ;;TELL APT TO TAKE MSG.
4593 022310 000413      BR    5$
4594 022312 017667 000004 000016 3$:      MOV   @4(SP),4$ ;;PUT MSG ADDR IN JSR LINKAGE
4595 022320 062766 000002 000004      ADD   #2,4(SP) ;;BUMP RETURN ADDRESS
4596 022326 016746 155444      MOV   177776,-(SP) ;;PUSH 177776 ON STACK
4597 022332 004767 177266      JSR   PC,$TYPE ;;CALL TYPE MACRO
4598 022336 000000      4$:      .WORD 0
4599 022340      5$:
    
```


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

4656 000001
 4657 000100
 4658 000040
 4659
 4660
 4661
 4662
 4663
 4664
 4665
 4666
 4667
 4668
 4669
 4670
 4671
 4672 022570
 4673 022570 010046
 4674 022572 010146
 4675 022574 010246
 4676 022576 010346
 4677 022600 010546
 4678 022602 012746 020200
 4679 022606 016605 000020
 4680 022612 100004
 4681 022614 005405
 4682 022616 112766 000055 000001
 4683 022624 016700 155702
 4684 022630 012703 023012
 4685 022634 060003
 4686 022636 112723 000040
 4687 022642 005002
 4688 022644 016001 023002
 4689 022650 160105
 4690 022652 002402
 4691 022654 005202
 4692 022656 000774
 4693 022660 060105
 4694 022662 005702
 4695 022664 001002
 4696 022666 105716
 4697 022670 100407
 4698 022672 106316
 4699 022674 103003
 4700 022676 116663 000001 177777
 4701 022704 052702 000060
 4702 022710 052702 000040
 4703 022714 110223
 4704 022716 005720
 4705 022720 020067 157110
 4706 022724 103746
 4707 022726 101002
 4708 022730 010502
 4709 022732 000764
 4710 022734 105726
 4711 022736 100003

APTENV=001
 APTSPool=100
 APTCSUP=040
 ;:*****

.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

\$TYPDS:
 MOV R0,-(SP) ;:PUSH R0 ON STACK
 MOV R1,-(SP) ;:PUSH R1 ON STACK
 MOV R2,-(SP) ;:PUSH R2 ON STACK
 MOV R3,-(SP) ;:PUSH R3 ON STACK
 MOV R5,-(SP) ;:PUSH R5 ON STACK
 MOV #20200,-(SP) ;:SET BLANK SWITCH AND SIGN
 MOV 20(SP),R5 ;:GET THE INPUT NUMBER
 BPL 1\$;:BR IF INPUT IS POS.
 NEG R5 ;:MAKE THE BINARY NUMBER POS.
 MOVB #'-,1(SP) ;:MAKE THE ASCII NUMBER NEG.
 1\$: MOV RELOC, R0 ;:GET RELOCATION FACTOR.
 MOV #SDBLK,R3 ;:SETUP THE OUTPUT POINTER
 ADD R0,R3 ;:ADD IN RELOCATION FACTOR.
 MOVB #' ,(R3)+ ;:SET THE FIRST CHARACTER TO A BLANK
 2\$: CLR R2 ;:CLEAR THE BCD NUMBER
 MOV \$DTBL(R0),R1 ;:GET THE CONSTANT
 3\$: SUB R1,R5 ;:FORM THIS BCD DIGIT
 BLT 4\$;:BR IF DONE
 INC R2 ;:INCREASE THE BCD DIGIT BY 1
 BR 3\$
 4\$: ADD R1,R5 ;:ADD BACK THE CONSTANT
 TST R2 ;:CHECK IF BCD DIGIT=0
 BNE 5\$;:FALL THROUGH IF 0
 TSTB (SP) ;:STILL DOING LEADING 0'S?
 BMI 7\$;:BR IF YES
 5\$: ASLB (SP) ;:MSD?
 BCC 6\$;:BR IF NO
 MOVB 1(SP),-1(R3) ;:YES--SET THE SIGN
 6\$: BIS #'0,R2 ;:MAKE THE BCD DIGIT ASCII
 7\$: BIS #' ,R2 ;:MAKE IT A SPACE IF NOT ALREADY A DIGIT
 MOVB R2,(R3)+ ;:PUT THIS CHARACTER IN THE OUTPUT BUFFER
 TST (R0)+ ;:JUST INCREMENTING
 CMP R0, .EIGHT ;:CHECK THE TABLE INDEX
 BLO 2\$;:GO DO THE NEXT DIGIT
 BHI 8\$;:GO TO EXIT
 MOV R5,R2 ;:GET THE LSD
 BR 6\$;:GO CHANGE TO ASCII
 8\$: TSTB (SP)+ ;:WAS THE LSD THE FIRST NON-ZERO?
 BPL 9\$;:BR IF NO

4712 022740 116663 177777 177776
 4713 022746 105013
 4714 022750 012605
 4715 022752 012603
 4716 022754 012602
 4717 022756 012601
 4718 022760 012600
 4719 022762 004567 176606
 4720 022766 023012
 4721 022770 016666 000002 000004
 4722 022776 012616
 4723 023000 000002
 4724 023002 023420
 4725 023004 001750
 4726 023006 000144
 4727 023010 000012
 4728 023012 000004
 4729
 4730
 4731
 4732
 4733
 4734
 4735
 4736
 4737
 4738
 4739
 4740
 4741
 4742
 4743
 4744
 4745
 4746
 4747
 4748
 4749
 4750
 4751
 4752
 4753
 4754 023022 017646 000000
 4755 023026 116667 000001 000213
 4756 023034 112667 000211
 4757 023040 062716 000002
 4758 023044 000406
 4759 023046 112767 000001 000173
 4760 023054 112767 000006 000167
 4761 023062 112767 000005 000156
 4762 023070 010346
 4763 023072 010446
 4764 023074 010546
 4765 023076 116704 000147
 4766 023102 005404
 4767 023104 062704 000006

```

9$:  MOV      -1(SP),-2(R3)  ;;YES--SET THE SIGN FOR TYPING
     CLR      (R3)          ;;SET THE TERMINATOR
     MOV      (SP)+,R5      ;;POP STACK INTO R5
     MOV      (SP)+,R3      ;;POP STACK INTO R3
     MOV      (SP)+,R2      ;;POP STACK INTO R2
     MOV      (SP)+,R1      ;;POP STACK INTO R1
     MOV      (SP)+,R0      ;;POP STACK INTO R0
     JSR      R5, $PRINT    ;GO PRINT OUT THE FOLLOWING MESSAGE.
     .WORD    $DBLK         ;ADDRESS OF MESSAGE TO BE TYPED
     MOV      2(SP),4(SP)   ;;ADJUST THE STACK
     MOV      (SP)+,(SP)
     RTI
$DTBL: 10000.
       1000.
       100.
       10.
$DBLK: .BLKW 4
.SBTTL BINARY TO OCTAL (ASCII) AND TYPE

*****
*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
*OCTAL (ASCII) NUMBER AND TYPE IT.
*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
*CALL:
*      MOV      NUM,-(SP)    ;;NUMBER TO BE TYPED
*      TYPOS    ;;CALL FOR TYPEOUT
*      .BYTE   N            ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
*      .BYTE   M            ;;M=1 OR 0
*                               ;;1=TYPE LEADING ZEROS
*                               ;;0=SUPPRESS LEADING ZEROS
*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
*$TYPOS OR $TYPOC
*CALL:
*      MOV      NUM,-(SP)    ;;NUMBER TO BE TYPED
*      TYPON    ;;CALL FOR TYPEOUT
*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
*CALL:
*      MOV      NUM,-(SP)    ;;NUMBER TO BE TYPED
*      TYPOC    ;;CALL FOR TYPEOUT
$TYPOS: MOV      @(SP),-(SP)  ;;PICKUP THE MODE
        MOV      1(SP),$OFILL ;;LOAD ZERO FILL SWITCH
        MOV      (SP)+,$OMODE+1 ;;NUMBER OF DIGITS TO TYPE
        ADD      #2,(SP)      ;;ADJUST RETURN ADDRESS
        BR      $TYPON
$TYPOC: MOV      #1,$OFILL    ;;SET THE ZERO FILL SWITCH
        MOV      #6,$OMODE+1 ;;SET FOR SIX(6) DIGITS
$TYPON: MOV      #5,$OCNT     ;;SET THE ITERATION COUNT
        MOV      R3,-(SP)     ;;SAVE R3
        MOV      R4,-(SP)     ;;SAVE R4
        MOV      R5,-(SP)     ;;SAVE R5
        MOV      $OMODE+1,R4  ;;GET THE NUMBER OF DIGITS TO TYPE
        NEG      R4
        ADD      #6,R4        ;;SUBTRACT IT FOR MAX. ALLOWED
  
```

```

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



```

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

```

4880
4881
4882
4883
4884
4885 023472 005015 052113 030461
4886 023500 024040 042515 047515
4887 023506 054522 046440 047101
4888 023514 043501 046505 047105
4889 023522 024524 040440 040526
4890 023530 046111 041101 042514
4891 023536 000
4892 023537 015 031012 020062
4893 023544 044502 020124 042101
4894 023552 020122 053101 044501
4895 023560 006514 000012
4896 023564 005015 042515 047515
4897 023572 054522 046440 050101
4898 023600 000072
4899 023602 005015 040520 044522
4900 023610 054524 046440 046505
4901 023616 051117 020131 040515
4902 023624 035120 000
4903 023627 015 043012 047522
4904 023634 020115 000
4905 023637 040 047524 000040
4906 023644 005015 047111 052523
4907 023652 043106 041511 042511
4908 023660 052116 046440 046505
4909 023666 051117 027131 027056
4910 023674 044506 051522 020124
4911 023702 033061 020113 047516
4912 023710 020124 046101 020114
4913 023716 044124 051105 020505
4914 023724 000
4915 023725 015 047012 020117
4916 023732 040520 044522 054524
4917 023740 051040 043505 051511
4918 023746 042524 051522 043040
4919 023754 052517 042116 000
4920 023761 015 051012 051505
4921 023766 040524 052122 047111
4922 023774 020107 043101 042524
4923 024002 020122 020101 047520
4924 024010 042527 020122 040506
4925 024016 046111 051125 006505
4926 024024 000012
4927 024026 005015 047516 050040
4928 024034 051101 052111 020131
4929 024042 051105 047522 051522
4930 024050 043040 052517 042116
4931 024056 047440 020116 042515
4932 024064 047515 054522 051440
4933 024072 040503 006516 000012
4934 024100 005015 051120 043517
4935 024106 040522 020115 047516

.SBTTL STANDARD PROGRAM MESSAGES
:*****
:VARIOUS MESSAGE PRINTOUTS USED THRUOUT
:THE PROGRAM
:*****
MMAMES: .ASCIZ <15><12>'KT11 (MEMORY MANAGEMENT) AVAILABLE'

AVAL22: .ASCIZ <15><12>'22 BIT ADR AVAIL'<15><12>

MEMMES: .ASCIZ <15><12>'MEMORY MAP:'

MTMAP: .ASCIZ <15><12>'PARITY MEMORY MAP:'

FROM: .ASCIZ <15><12>'FROM '
TO: .ASCIZ ' TO '
INSUFF: .ASCIZ <15><12>'INSUFFICIENT MEMORY...FIRST 16K NOT ALL THERE!'

MTR: .ASCIZ <15><12>'NO PARITY REGISTERS FOUND'

PWRMSG: .ASCIZ <15><12>'RESTARTING AFTER A POWER FAILURE'<15><12>

NOPE: .ASCIZ <15><12>'NO PARITY ERRORS FOUND ON MEMORY SCAN'<15><12>

PROREL: .ASCII <15><12>'PROGRAM NOW RESIDES BACK AT 0 TO 8K'

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

4992 024564 052040 040510 020116
4993 024572 047117 020105 040520
4994 024600 044522 054524 042440
4995 024606 051122 051117 043040
4996 024614 052517 042116 000056
4997 024622 005015 041523 047101
4998 024630 044516 043516 046440
4999 024636 046505 051117 020131
5000 024644 047506 020122 040502
5001 024652 020104 040520 044522
5002 024660 054524 000056
5003 024664 005015 040520 044522
5004 024672 054524 042440 051122
5005 024700 051117 053440 046111
5006 024706 020114 047516 020124
5007 024714 046103 040505 027122
5008 024722 000
5009 024723 015 047012 020117
5010 024730 042515 047515 054522
5011 024736 052040 051505 042524
5012 024744 027104 000
5013 024747 015 051412 044513
5014 024754 050120 047111 020107
5015 024762 042524 052123 021440
5016 024770 000
5017 024771 377 000377
5018
5019
5020
5021
5022
5023 024774 040520 044522 054524
5024 025002 051040 043505 051511
5025 025010 042524 020122 040504
5026 025016 040524 042440 051122
5027 025024 051117 000056
5028 025030 042101 051104 051505
5029 025036 020123 042524 052123
5030 025044 042440 051122 051117
5031 025052 052050 052123 026461
5032 025060 024465 000056
5033 025064 047503 051516 040524
5034 025072 052116 042040 052101
5035 025100 020101 051105 047522
5036 025106 024122 051524 033124
5037 025114 030455 024460 000056
5038 025122 047522 040524 044524
5039 025130 043516 041040 052111
5040 025136 042440 051122 051117
5041 025144 052050 052123 030461
5042 025152 030455 024462 000056
5043 025160 047515 020123 042522
5044 025166 051106 051505 020110
5045 025174 042524 052123 042440
5046 025202 051122 051117 024040
5047 025210 051524 020124 032062

SCANM: .ASCIZ <15><12>'SCANNING MEMORY FOR BAD PARITY.'

PEWNC: .ASCIZ <15><12>'PARITY ERROR WILL NOT CLEAR.'

NOMTST: .ASCIZ <15><12>'NO MEMORY TESTED.'

SKPMES: .ASCIZ <15><12>'SKIPPING TEST #'

FILL2: .ASCIZ <377><377>

.SBTTL ERROR REPORTING MESSAGES AND TABLES.
:*****
:* MESSAGE BLOCK FOR ERROR TABLE TYPEOUTS
:*****

DM1: .ASCIZ 'PARITY REGISTER DATA ERROR.'

DM2: .ASCIZ 'ADDRESS TEST ERROR(TST1-5).'

DM4: .ASCIZ 'CONSTANT DATA ERROR(TST6-10).'

DM5: .ASCIZ 'ROTATING BIT ERROR(TST11-12).'

DM6: .ASCIZ 'MOS REFRESH TEST ERROR (TST 24-25).'

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

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

5155
5156
5157
5158
5159

 :DATA COLUMN HEADINGS

```

5216 027006 000
5217 027007 122 043505 020011 DH30: .ASCIZ 'REG WAS MA WAS'
5218 027014 053440 051501 020011
5219 027022 046440 004501 020040
5220 027030 020040 040527 000123
5221 027036 027526 041520 050011 DH31: .ASCIZ 'V/PC P/PC MA S/B WAS'
5222 027044 050057 004503 020040
5223 027052 040515 020011 020040
5224 027060 051440 041057 020040
5225 027066 040527 000123
    
```

```

*****
;* DATA FORMAT TABLE FOR ERROR PRINTOUT.
*****
    
```

```

5229
5230 027072 000 377 000 DF1: .BYTE 0,-1,0,0
5231 027075 000
5232 027076 000 377 377 DF2: .BYTE 0,-1,-1,0,0
5233 027101 000 000
5234 027103 000 377 377 DF3: .BYTE 0,-1,-1,-2,-2
5235 027106 376 376
5236 027110 000 377 377 DF14: .BYTE 0,-1,-1,-1,0,0
5237 027113 377 000 000
5238 027116 000 377 000 DF21: .BYTE 0,-1,0,-1,0,0
5239 027121 377 000 000
5240 027124 377 000 377 DF30: .BYTE -1,0,-1,-2
5241 027127 376
    
```

.EVEN

```

5242
5243
5244 032000 . = 32000
    
```

```

;THE LOADERS ARE SAVE HERE TO END OF 8K
;1536. WORDS
    
```

```

5245
5246 000001 .END
5247
    
```


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

\$MADR2	001232	406#													
\$MADR3	001236	409#													
\$MADR4	001242	412#													
\$MAIL	001174	375#	454	458	1175	1193	4013	4067	4493						
\$MAMS1	001224	396#													
\$MAMS2	001230	404#													
\$MAMS3	001234	407#													
\$MAMS4	001240	410#													
\$MBADR	001322	454#													
\$MFLG	022564	4570*	4576	4626*	4651#										
\$MNEW	021407	4229	4391#												
\$MSGAD	001210	382#	4586*	4589	4637*										
\$MSGLG	001212	383#	4591*	4636*											
\$MSGTY	001174	376#	4584	4592*	4619	4623*	4634	4638*							
\$MSWR	021376	4220	4389#												
\$MTYP1	001225	397#													
\$MTYP2	001231	405#													
\$MTYP3	001235	408#													
\$MTYP4	001241	411#													
\$MXCNT	017652	4011	4021#												
\$NULL	001144	356#	4520	4566											
\$NWTST=	000001	1824#	1826	1867#	1869	1906#	1908	1946#	1948	1980#	1982	2040#	2042	2043	
		2058#	2060	2089#	2091	2111#	2132#	2153#	2155	2350#	2385#	2387	2430#	2432	
		2475#	2477	2521#	2523	2568#	2570	2613#	2615	2660#	2662	2748#	2750	2788#	
		4761*	4791*	4804#											
\$OCNT	023246	4756*	4760*	4765	4768*	4779*	4806#								
\$OMODE	023250	3973	3991	3999	4009	4018#									
\$OVER	017636	379#	1175*	2827	2831	2858*	2859*	2868	2893	4005	4022				
\$PASS	001202	456#													
\$PASTM	001326	315	1188	1248	1266	1386	1390	1402	1439	1539	1548	1585	1693	1728	
\$PRINT	021574	1781	1786	2844	2866	2875	3306	3487	3505	3619	3663	3723	3809	3862	
		3890	4055	4064	4106	4131	4133	4137	4139	4184	4190	4216	4219	4228	
		4242	4255	4276	4337	4344	4350	4355	4361	4363	4367	4371	4376	4447	
		4449	4459#	4513	4719	4789									
\$PWRAD	000702	318#													
\$PWDRN	000540	285#	313	1156											
\$PWRMG	000676	316#													
\$PWRUP	000612	295	301#												
\$QUES	001170	367#	4095	4277	4368	4387	4450	4453	4566						
\$RDCHR	020736	4290#	4330												
\$RDLIN	021066	4320#	4418												
\$RDOCT	021420	1702	1736	1794	4407#										
\$RDSZ =	000C10	4313#													
\$SAVR6	000712	294*	302	303*	304*	322#									
\$SCOPE	017354	1834	1877	1916	1956	1990	2051	2063	2094	2115	2136	2164	2354	2411	
		2456	2501	2549	2594	2639	2685	2762	2792	2820	3964#	4077	4090	4200	
\$SETUP=	000130	188#	1155	1156	1158	1159	1186	1190	2857	3965	4043				
		4393													
\$STUP =	177777	188#													
\$SVLAD	017602	3981	4012#												
\$SVPC =	000010	209#	214												
\$SWR =	167400	1#	12	17	18	19	20	21	22	23	24	319	364	365	
		366	1159	1835	1878	1917	1957	1991	2052	2064	2095	2116	2137	2165	
		2355	2412	2457	2502	2550	2595	2640	2686	2763	2793	2852	2857	2878	
		2892	2893	3956	3957	3958	3959	3960	3972	3984	3986	3987	3992	3993	
		3994	4001	4002	4003	4015	4018	4021	4034	4035	4036	4037	4038	4053	

.STACK	002002	623#	1807			
.TSTMA	002010	626#	1810	2026	3020	3105
.TST32	002036	637#				
.SASTA=	***** U	4570	4573			
.SX =	001320	442#	447			

.TM7	2058#	2060
.SACT1	1#	205
.SAPT8	1#	371#
.SAPTH	1#	437
.SAPTY	1#	4566
.SASTA	1#	459
.SCATC	1#	188
.SCMTA	1#	323
.SDB2D	1#	
.SDB20	1#	
.SDIV	1#	
.SEOP	1#	2846
.SERRO	1#	4028
.SERRT	1#	4095
.SMULT	1#	
.SPOWE	1#	281
.SRAND	1#	
.SRDDE	1#	
.SRDOC	1#	4393
.SREAD	1#	4195
.SR2AZ	1#	
.SSAVE	1#	
.SSB2D	1#	
.SSB20	1#	
.SSCOP	1#	3950
.SSIZE	1#	
.SSUPR	1#	
.STRAP	1#	
.STYPB	1#	
.STYPD	1#	4659
.STYPE	1#	4470
.STYPO	1#	4729
.\$40CA	1#	
.1170	1#	

. ABS. 032000 000

ERRORS DETECTED: 0

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