M11/ME11

07124K MEMORY EXERCISER 16K VERIFY

MD-11-DZQMC-B

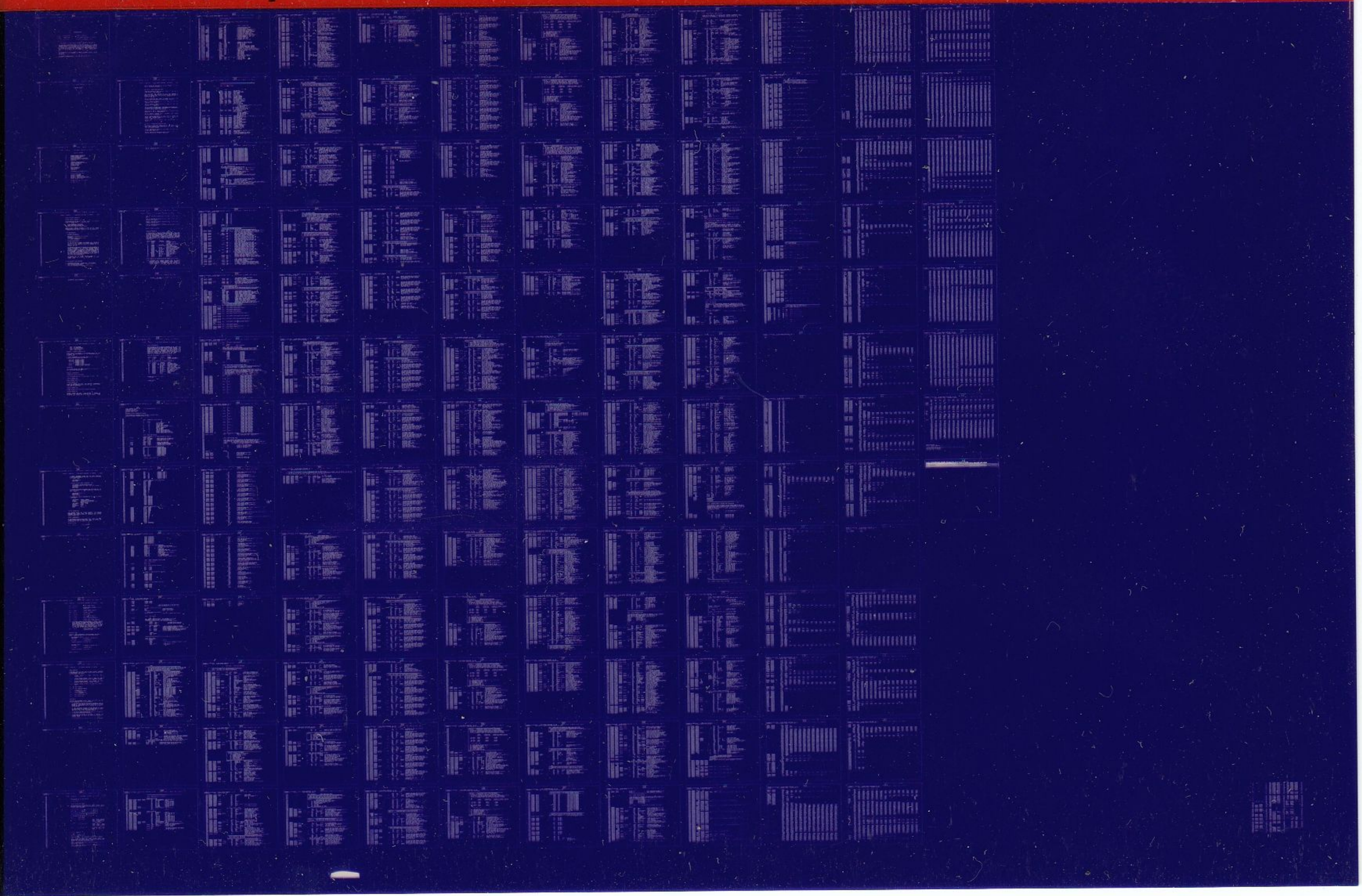
EP-DZQMC-B-DL-A

OCT 1976

FICHE 1 OF 1

digital

Made In U.S.A.



.REM %

IDENTIFICATION
==============

PRODUCT CODE: MAINDEC-11-DZQMC-B-D

PRODUCT NAME: D-124K MEMORY EXERCISER, 16K VERSION.

MAINTAINER: DIAGNOSTIC ENGINEERING, CC301

REVISION HISTORY

REVISION A: MAY 1975

REVISION B: OCTOBER 1975

TABLE OF CONTENTS

1.0 GENERAL PROGRAM INFORMATION.

1.1 PROGRAM PURPOSE (ABSTRACT)

THIS PROGRAM HAS THE ABILITY TO TEST MEMORY FROM ADDRESS 000000 TO ADDRESS 757777. IT DOES SO USING:

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

THERE IS ALSO A SPECIAL ROUTINE TO TYPE OUT ALL UNIBUS ADDRESS RANGES WHICH DO NOT TIMEOUT, AS WELL AS TWO(2) TOGGLE IN ADDRESS TESTS PROVIDED IN SECTION 6.1 OF THIS DOCUMENT.

1.2 SYSTEM REQUIREMENTS

A. HARDWARE REQUIREMENTS

PDP11 FAMILY PROCESSOR WITH A MINIMUM OF 16K OF MEMORY.
OPTIONAL...
M7259 PARITY MEMORY CONTROL MODULE.
KT11 MEMORY MANAGEMENT.

B. SOFTWARE REQUIREMENTS

THE SMALLEST UNIT OF MEMORY THIS PROGRAM WILL RECOGNIZE IS 4K. IF ANY ADDRESS IN A 4K BANK CAUSES A TIME OUT TRAP, THAT ENTIRE BANK OF MEMORY IS IGNORED BY THE PROGRAM.

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.

1.3 RELATED DOCUMENTS AND STANDARDS

A. PROGRAMMING PRACTICES - DOCUMENT NO. 175-003-009-01
B. PDP-11 MAINDEC SYSMAC PACKAGE - MAINDEC-11-DZQAC-B-D
C. PDP11/40 PROCESSOR HANDBOOK
D. MF11-U/UP CORE MEMORY SYSTEM MAINTENANCE MANUAL

```
                                        DOCUMENT NO.  DEC-11-HMFMA-B-D
                                    E.  APPLICABLE CIRCUIT SCHEMATICS:
```

G235 - 16K X-Y DRIVE
G114 - 16K SENSE/INHIBIT
M8293 - 16K UNIBUS TIMING
M7259 - PARITY CONTROL

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.

IF MEMORY MANAGEMENT IS TO BE USED, THEN THE KT11 DIAGNOSTIC SHOULD ALSO BE RUN BEFORE THIS PROGRAM.

PDP-11/05 - MAINDEC-11-DZQKC
PDP-11/20 - MAINDEC-11-DZQKC
PDP-11/40 - MAINDEC-11-DBQEA
OR MAINDEC-11-DCQKC
PDP-11/45 - MAINDEC-11-DCQKC
KT11-C - MAINDEC-11-DCKTA THRU DCKTF
KT11-D - MAINDEC-11-DBKTA THRU DBKTF

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 ANY STANDARD ABSOLUTE LOADER.

2.1.2 STARTING ADDRESS 200:

NORMAL PROGRAM EXECUTION.

2.1.3 STARTING ADDRESS 204:

ALLOWS THE OPERATOR TO INPUT, VIA TELETYPE CONVERSATION, FIRST AND LAST ADDRESSES TO BE EXERCISED, AND A DATA PATTERN TO BE USED IN TESTS 6 AND 7.

2.1.4 STARTING ADDRESS 210:

RESTART PROGRAM USING PREVIOUSLY SELECTED PARAMETERS.

2.1.5 STARTING ADDRESS 214:

RESTORE LOADERS AND HALT. THIS ROUTINE IS CAPABLE OF RELOCATING THE PROGRAM BACK TO BANKS 0 AND 1 IF THE PROGRAM WAS HALTED WHILE RUNNING THE TOP TWO BANKS OF MEMORY. THERE

202 ARE SPECIAL PROCEDURES REQUIRED FOR THIS SITUATION.

A. IF MEMORY ADDRESSES 0-1000 HAVE NOT BEEN EXERCISED, EITHER THROUGH PARAMETER SELECTION (SA=204) OR BY RUNNING WITH SW05=1, THEN:

LOAD ADDRESS 214,
PRESS START.

B. IF RUNNING WITHOUT MEMORY MANAGEMENT, THEN:

LOAD ADDRESS <214+RELOCATION FACTOR>
(RELOCATION FACTOR IS TYPED WHEN THE PROGRAM IS RELOCATED),
PRESS START.

C. IF RUNNING WITH MEMORY MANAGEMENT AND THE UNIBUS HAS NOT BEEN INITIALIZED (VIA RESET INSTRUCTION, START SWITCH, ETC.), THEN:

LOAD ADDRESS 777707 (PC)
DEPOSIT 214
PRESS CONTINUE

D. IF RUNNING WITH MEMORY MANAGEMENT AND THE UNIBUS HAS BEEN INITIALIZED:

LOAD ADDRESS	772340 (KIPAR0)
DEPOSIT	<(RELOCATION FACTOR)/100>
(EXAMPLE:	RELOCATION FACTOR=540000, THEN
	DEPOSIT 005400)
LOAD ADDRESS	777572 (SR0)
DEPOSIT	000001
LOAD ADDRESS	777707 (PC)
DEPOSIT	214
PRESS	CONTINUE

2.1.6 STARTING ADDRESS 220:

BYTE ADDRESS MEMORY MAP TYPEOUT ROUTINE. THIS ROUTINE PERFORMS DATI, DATIP, DATO, AND DATOB ON ALL POSSIBLE ADDRESSES, AND TYPES THE RANGES OF ADDRESSES WHICH DO NOT CAUSE A TIMEOUT TRAP.

2.2 SPECIAL ENVIRONMENTS

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

2.3 PROGRAM OPTIONS

SW15 = 1 OR UP.... HALT ON ERROR

```
259
260                                    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

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

NOTE: WITH PARITY ERROR DETECTION ENABLED, A MEMORY FAILURE WHILE RUNNING THE WORSE CASE NOISE TESTS (NON-PARITY) CAN CAUSE A PARITY ERROR. THE ERROR PRINTOUT ON A PARITY ERROR DOES NOT TYPE THE GOOD DATA. THUS A BIT DROP OR PICKUP WILL NOT BE TYPED AS SUCH. IT IS BEST TO RUN THE PROGRAM FOR 1 PASS WITH PARITY DISABLED, THEN, RESTART THE PROGRAM WITH PARITY ENABLED.

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 900NS MEMORYS ARE:

A. FOR NON-PARITY MEMORY

FIRST PASS:	45 SECONDS + 3 SECONDS PER 4K
FULL PASS:	3 MINUTES PER 4K
ITERATION INHIBITED:	45 SECONDS PER 4K

B. FOR PARITY MEMORY

FIRST PASS:	45 SECONDS +10 SECONDS PER 4K
FULL PASS:	4 MINUTES PER 4K
ITERATION INHIBITED:	1 MINUTE PER 4K

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

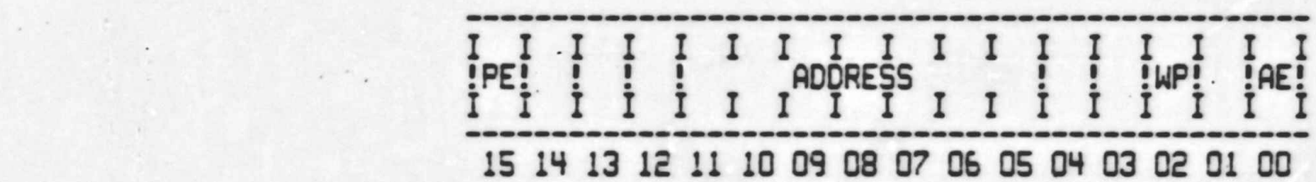
373 CONTROL ROUTINES, FAILURES TO FIND A MEANINGFUL MAP.

4.0 PERFORMANCE AND PROGRESS REPORTS

NOT APPLICABLE

5.0 DEVICE INFORMATION TABLES

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



BIT ASSIGNMENTS ARE DEFINED AS FOLLOWS:

BIT15	PARITY ERROR	
BITS 11-5	ERROR ADDRESS	HIGH ORDER ADDRESS BITS OF ADDRESS OF PARITY ERROR (BITS 17-11 OF ADDRESS)
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

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 4K 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 CORE 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.

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 WRITE A PATTERN WHICH COMPLEMENTS WHEN ADDRESS BIT 1 XOR ADDRESS BIT 8 COMPLEMENTS. THIS CREATES A "CHECKERBOARD" PATTERN IN CERTAIN MEMORY ARCHITECTURES.

TEST 14,15,16, AND 17 WRITE A PATTERN WHICH COMPLEMENTS WHEN ADDRESS BIT 3 XOR BIT 9 COMPLEMENTS.

TEST 20 WRITES A PATTERN WHICH COMPLEMENTS WHEN ADDRESS BIT 8 XOR BIT 13 COMPLEMENTS.

TEST 21 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 22 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 23 EXECUTES AN INSTRUCTION WHICH DOES A DATI AND A DATO ON THE MEMORY UNDER TEST.

TEST 24 EXECUTES AN INSTRUCTION WHICH DOES A DATI AND A DATOB ON THE LOW BYTE OF MEMORY UNDER TEST.

```
488                                     TEST 25  EXECUTES AN INSTRUCTION WHICH  DOES  A  DATI  AND  A
489                                     DATOB ON THE HIGH BYTE.
```

TEST 26 EXECUTES AN INSTRUCTION WHICH DOES A DATIP AND A DATO.

TEST 27 EXECUTES AN INSTRUCTION WHICH DOES A DATIP AND A DATOB ON THE LOW BYTE.

TEST 30 EXECUTES AN INSTRUCTION WHICH DOES A DATIP AND A DATOB ON THE HIGH BYTE.

TEST 31 EXECUTES A SERIES OF INSTRUCTIONS DESIGNED TO STRESS OR "HEAT" THE MEMORY.

6.4 SPECIAL TOGGLE IN TESTS

6.4.1 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 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 #50,R0   ;GET FIRST ADDRESS
* 12     000050                     ;TO TEST
                                    ;(EXAMPLE START ADDRESS)
  14     010001        MOV R0,R1    ;SAVE IN R1
  16     020037    1$: CMP R0,@#SWR ;CHECK UPPER LIMIT
  20     177570                     ;(IN SWITCH REGISTER)
  22     001403        BEQ 2$       ;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
```

AFTER TOGGLING THE PROGRAM LA=10**SET UPPER LIMIT**, START

NOTES: THE UPPER LIMIT ADDRESS OBTAINED FROM THE SWITCH REGISTER MAY BE CHANGED DURING PROGRAM OPERATION. HOWEVER OCCASIONALLY THE PROGRAM MAY HALT BECAUSE OF 'SWITCH BOUNCE'. (THE BEST PROCEDURE WHEN CHANGING LIMITS IS TO STOP THE PROGRAM MAKE THE CHANGE AND

```
546
547                                     CONTINUE.)  THE LOWER LIMIT ADDRESS (12) MAY BE PATCHED
                                        TO ANY DESIRED ADDRESS.
```

6.4.2 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.

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

LOCATION	CONTENTS	MNEMONIC	COMMENT
12	100		;CHANGE LOWER LIMIT
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 FUNCTIONAL FLOW CHARTS
ATTACHED

8.0 PROGRAM LISTING
ATTACHED

```
591                                             %

                .TITLE  MAINDEC-11-DZQMC-B-D:   0-124K MEMORY EXERCISER, 16K VER
                ;*COPYRIGHT (C) 1975
                ;*DIGITAL EQUIPMENT CORP.
                ;*MAYNARD, MASS. 01754
                ;*
                ;*PROGRAM BY BRUCE BURGESS/KEN CHAPMAN
                ;*
                ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
                ;*PACKAGE (MAINDEC-11-DZQAC-B1),AUG 29,1975.
                ;*

604
605                                     .SBTTL  OPERATIONAL SWITCH SETTINGS
606                                     ;*
607                                     ;*      SWITCH                  USE
608                                     ;*      ------          --------------------
609                                     ;*        15            HALT ON ERROR
610                                     ;*        14            LOOP ON TEST
611                                     ;*        13            INHIBIT ERROR TYPEOUTS
612                                     ;*        12            INHIBIT KT11 (AT START TIME ONLY)
613                                     ;*        11            INHIBIT ITERATIONS
614                                     ;*        10            BELL ON ERROR
615                                     ;*         9            LOOP ON ERROR
616                                     ;*         8            LOOP ON TEST IN SWR<4:0>
617                                     ;*         7            INHIBIT PROGRAM RELOCATION
618                                     ;*         6            INHIBIT PARITY ERROR DETECTION
619                                     ;*         5            INHIBIT EXRERCISING VECTOR AREA.
620
621                                     .SBTTL  BASIC DEFINITIONS
622
623                                     ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
624             001100                  STACK=  1100
625                                     .EQUIV  EMT,ERROR       ;;BASIC DEFINITION OF ERROR CALL
626                                     .EQUIV  IOT,SCOPE       ;;BASIC DEFINITION OF SCOPE CALL
627             177776                  PS=     177776          ;;PROCESSOR STATUS WORD
628                                     .EQUIV  PS,PSW
629             177774                  STKLMT= 177774          ;;STACK LIMIT REGISTER
630             177772                  PIRQ=   177772          ;;PROGRAM INTERRUPT REQUEST REGISTER
631             177570                  DSWR=   177570          ;;HARDWARE SWITCH REGISTER
632             177570                  DDISP=  177570          ;;HARDWARE DISPLAY REGISTER
633
634                                     ;*GENERAL PURPOSE REGISTER DEFINITIONS
635             000000                  R0=     %0              ;;GENERAL REGISTER
636             000001                  R1=     %1              ;;GENERAL REGISTER
637             000002                  R2=     %2              ;;GENERAL REGISTER
638             000003                  R3=     %3              ;;GENERAL REGISTER
639             000004                  R4=     %4              ;;GENERAL REGISTER
640             000005                  R5=     %5              ;;GENERAL REGISTER
641             000006                  R6=     %6              ;;GENERAL REGISTER
642             000007                  R7=     %7              ;;GENERAL REGISTER
643                                     .EQUIV  R6,SP           ;;STACK POINTER
644                                     .EQUIV  R7,PC           ;;PROGRAM COUNTER
645
646                                     ;*PRIORITY LEVEL DEFINITIONS
```

```
                                PR0=    0               ;;PRIORITY LEVEL 0
                                PR1=    40              ;;PRIORITY LEVEL 1
                                PR2=    100             ;;PRIORITY LEVEL 2
                                PR3=    140             ;;PRIORITY LEVEL 3
                                PR4=    200             ;;PRIORITY LEVEL 4
                                PR5=    240             ;;PRIORITY LEVEL 5
                                PR6=    300             ;;PRIORITY LEVEL 6
                                PR7=    340             ;;PRIORITY LEVEL 7

                                ;*"SWITCH REGISTER" SWITCH DEFINITIONS
                                SW15=   100000
                                SW14=   40000
                                SW13=   20000
                                SW12=   10000
                                SW11=   4000
                                SW10=   2000
                                SW09=   1000
                                SW08=   400
                                SW07=   200
                                SW06=   100
                                SW05=   40
                                SW04=   20
                                SW03=   10
                                SW02=   4
                                SW01=   2
                                SW00=   1
                                .EQUIV  SW09,SW9
                                .EQUIV  SW08,SW8
                                .EQUIV  SW07,SW7
                                .EQUIV  SW06,SW6
                                .EQUIV  SW05,SW5
                                .EQUIV  SW04,SW4
                                .EQUIV  SW03,SW3
                                .EQUIV  SW02,SW2
                                .EQUIV  SW01,SW1
                                .EQUIV  SW00,SW0

                                ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
                        100000  BIT15=  100000
                        040000  BIT14=  40000
                        020000  BIT13=  20000
                        010000  BIT12=  10000
                        004000  BIT11=  4000
                        002000  BIT10=  2000
                        001000  BIT09=  1000
                        000400  BIT08=  400
                        000200  BIT07=  200
                        000100  BIT06=  100
                        000040  BIT05=  40
                        000020  BIT04=  20
                        000010  BIT03=  10
                        000004  BIT02=  4
                        000002  BIT01=  2
                        000001  BIT00=  1
                                .EQUIV  BIT09,BIT9
                                .EQUIV  BIT08,BIT8
```

```
                        .EQUIV  BIT07,BIT7
                        .EQUIV  BIT06,BIT6
                        .EQUIV  BIT05,BIT5
                        .EQUIV  BIT04,BIT4
                        .EQUIV  BIT03,BIT3
                        .EQUIV  BIT02,BIT2
                        .EQUIV  BIT01,BIT1
                        .EQUIV  BIT00,BIT0

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


                        .SBTTL  MEMORY MANAGEMENT DEFINITIONS

                        ;*KT11 VECTOR ADDRESS

000250                  MMVEC=  250

                        ;*KT11 STATUS REGISTER ADDRESSES

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

                        ;*KERNAL "I" PAGE DESCRIPTOR REGISTERS

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

                        ;*KERNAL "I" PAGE ADDRESS REGISTERS

172340                  KIPAR0= 172340
172342                  KIPAR1= 172342
172344                  KIPAR2= 172344
172346                  KIPAR3= 172346
172350                  KIPAR4= 172350
```

```
                    172352                  KIPAR5= 172352
                    172354                  KIPAR6= 172354
                    172356                  KIPAR7= 172356

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

                                            ;*  PARITY MEMORY DEFINITIONS.
                    000004                  WWP=4                           ;WRITE WRONG PARITY
                    000001                  AE=1                            ;PARITY ACTION ENABLE
                    000114                  PARVEC=114                      ;PARITY TRAP VECTOR

                                            ;*  MISCELLANEOUS ASSIGNMENTS
                    017777                  MASK4K= 17777                   ;MASK FOR 4K ADDRESS BANK BOUNDRY.


                                            .SBTTL  TRAP CATCHER

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

                                            .SBTTL  STARTING ADDRESS(ES)
000200  000137  002624                         JMP     @#START         ;;JUMP TO STARTING ADDRESS OF PROGRAM
000204  000167  002422                         JMP     SELECT          ;STARTING ADDRESS TO ALLOW THE OPERATOR TO
                                                                        ;SELECT VARIOUS PARAMETERS.
000210  000167  000064                         JMP     RESTAR          ;RESTART ADDRESS, USING PREVIOUS PARAMETERS.
000214  000167  000064                         JMP     RESTOR          ;RESTORE LOADERS TO END OF MEMORY AND HALT.
000220  000167  003272                         JMP     TIMOUT          ;TYPE OUT MEMORY MAP, BYTE BY BYTE.

                    000004                          .=ERRVEC
000004  024636                                  .WORD   ERRTRP
000006  000000                                  .WORD   0

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

                                            .SBTTL  ACT11 HOOKS
                                            ;HOOKS REQUIRED BY ACT11
                    000010                          $SVPC=.                 ;SAVE PC
                    000046                          .=46
000046  014546                                  $ENDAD                  ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
                    000052                          .=52
000052  040000                                  .WORD   BIT14           ;;2)SET LOC.52 TO BIT14
                    000010                          .=$SVPC                 ;; RESTORE PC
```

```
  807           000300                          .=300
  808                                 ;*****************************************************************
  809                                 ;*  THE FOLLOWING ROUTINES ARE LOCATED IN THE VECTOR AREA (0-1000) SO THAT
  810                                 ;*   THEY CAN BE PROTECTED BY SELECTING SW05 (SEE DOCUMENT FOR USE OF SW05).
  811                                 ;*   THE CODE CAN ALSO BE RUN FROM ANY BANK OF MEMORY, ASSUMING MEMORY
  812                                 ;*   MANAGEMENT IS DISABLED BY "CONSOLE START".
  813                                 ;*****************************************************************
  814   000300  005005                RESTAR: CLR     R5              ;CLEAR FLAG TO INDICATE RESTART.
  815   000302  000401                        BR      REST1           ;GO RESTORE PROGRAM BEFORE RESTARTING.
  816   000304  010705                RESTOR: MOV     PC,     R5      ;PUT DATA INTO FLAG FOR RESTORE.
  817   000306  012706  001100        REST1:  MOV     #STACK, SP      ;SET UP THE STACK POINTER.
  818   000312  005767  001204                TST     MEMMAP          ;CHECK IF THE MEMORY HAS BEEN MAPPED.
  819   000316  001002                        BNE     REST2           ;BR IF MEMORY MAPPED.
  820   000320  000167  002314                JMP     START           ;GO START
  821   000324  005767  000256        REST2:  TST     MMAVA           ;CHECK IF MEM MGMT AVAILABLE.
  822   000330  001470                        BEQ     10$             ;BR IF NO MEM MGMT.
  823   000332  032737  000001  177572        BIT     #BIT0,  @#SR0   ;CHECK IF MEM MGMT ACTIVE.
  824   000340  001034                        BNE     2$              ;BR IF MEM MGMT ALREADY SET UP.
  825   000342  012700  172300                MOV     #KIPDR0,R0      ;POINT TO FIRST MEM MGMT DDATA REG.
  826   000346  012701  000010                MOV     #8.,    R1      ;SET UP COUNTER.
  827   000352  012720  077406        1$:     MOV     #077406,(R0)+   ;MAP FIRST 28K 1-FOR-1.
  828   000356  005301                        DEC     R1              ;COUNT REGESTERS.
  829   000360  001374                        BNE     1$              ;BR IF MORE REG.
  830   000362  012700  172340                MOV     #KIPAR0,R0      ;POINT TO FIRST MEM MGMT ADDRESS REG.
  831   000366  005020                        CLR     (R0)+           ;PAR0 MAPPED INTO BANK0.
  832   000370  012720  000200                MOV     #200,   (R0)+   ;PAR1 MAPPED INTO BANK1.
  833   000374  012720  000400                MOV     #400,   (R0)+   ;PAR2 MAPPED INTO BANK2.
  834   000400  012720  000600                MOV     #600,   (R0)+   ;PAR3 MAPPED INTO BANK3.
  835   000404  012720  001000                MOV     #1000,  (R0)+   ;PAR4 MAPPED INTO BANK4.
  836   000410  012720  001200                MOV     #1200,  (R0)+   ;PAR5 MAPPED INTO BANK5.
  837   000414  012720  001400                MOV     #1400,  (R0)+   ;PAR6 MAPPED INTO BANK6.
  838   000420  012720  007600                MOV     #7600,  (R0)+   ;PAR7 MAPPED INTO BANK37.
  839   000424  012737  000001  177572        MOV     #BIT0,  @#SR0   ;ENABLE MEM MGMT.
  840   000432  005000                2$:     CLR     R0              ;INIT TEMP PAR REG.
  841   000434  016701  000142                MOV     PRGMAP, R1      ;GET THE PROGRAM MAP...LO 64K.
  842   000440  016702  000140                MOV     PRGMAP+2,R2     ;...HI 64K.
  843   000444  006202                3$:     ASR     R2              ;SHIFT THE MAP POINTER...HI
  844   000446  006001                        ROR     R1              ;...LO.
  845   000450  103404                        BCS     4$              ;BR WHEN FIRST BANK FOUND.
  846   000452  062700  000200                ADD     #200,   R0      ;UPDATE TMP PAR TO NEXT BANK.
  847   000456  100372                        BPL     3$              ;BR IF MORE.
  848   000460  000000                        HALT                    ;FATAL ERROR!!!  MAP EMPTY?
  849   000462  010037  172340        4$:     MOV     R0,     @#KIPAR0 ;PUT TEMP PAR INTO FIRST PAR.
  850   000466  000137  000472                JMP     @#5$            ;JUMP INTO PROGRAM IF NOT THERE ALREADY.
  851   000472  062700  000200        5$:     ADD     #200,   R0      ;KEEP UPDATING TEMP PAR REG.
  852   000476  006202                        ASR     R2              ;SHIFT POINTER...HI
  853   000500  006001                        ROR     R1              ;...LO
  854   000502  103373                        BCC     5$              ;BR IF TOP BANK NOT YET FOUND.
  855   000504  010037  172342                MOV     R0,     @#KIPAR1 ;SET UP SECOND PROGRAM ANK POINTER.
  856   000510  000410                        BR      20$             ;BR TO RELOCATE SECTION.
  857   000512  016700  000062        10$:    MOV     RELOCF, R0      ;GET RELOCATION FACTOR.
  858   000516  062700  001100                ADD     #STACK, R0      ;SET UP STACK POINTER.
  859   000522  010006                        MOV     R0,     SP      ;SET STACK TO RELOCATE PROGRAM.
  860   000524  062700  177432                ADD     #20$-STACK,R0   ;ADJUST R0 TO RELOCATED "20$" ADDRESS.
  861   000530  000110                        JMP     (R0)            ;GO TO "20$" (RELOCATED).
  862   000532  022767  000003  000042 20$:   CMP     #3,     PRGMAP  ;CHECK IF PROGRAM IS IN BANKS 0 AND 1.
```

```
  863  000540  001402                        BEQ     21$             ;BR IF IN BANKS 0 AND 1.
  864  000542  004767  016616                JSR     PC,     RELO    ;RELOCATE THE PROGRAM BACK TO BANKS 0 AND 1.
  865  000546  005705              21$:      TST     R5              ;CHECK RESTART/RESTORE FLAG.
  866  000550  001006                        BNE     22$             ;BR IF RESTORE.
  867  000552  005067  000410                CLR     $TIMES          ;CLEAN UP BEFORE STARTING.
  868  000556  105067  000320                CLRB    $TSTNM
  869  000562  000167  004576                JMP     START1          ;RESTART WITH PREVIOUSLY SELECTED PARAMETERS.
  870  000566  004767  017000      22$:      JSR     PC,     RESLDR  ;RESTORE THE LOADERS TO THE "TOP" OF MEMORY.
  871  000572  000000                        HALT                    ;HALT AFTER RESTORING THE LOADERS.
  872  000574  000167  002040                JMP     STARTA          ;CONTINUE WILL RESTART THE PROGRAM.
  873                              ;* THE FOLLOWING LOCATIONS ARE USED BY THE ABOVE ROUTINE AND MUST BE LOCATED
  874                              ;*  BELOW 1000 TO INSURE CORRECT OPERATION UNDER THE WIDEST VARIETY OF
  875                              ;*  CIRCUMSTANCES.
  876  000600  000000              RELOCF: .WORD   0               ;CONTAINS RELOCATION FACTOR (NO MEM MGMT)
  877  000602  000000  000000      PRGMAP: .WORD   0,0             ;PROGRAM MAP - WHERE THE PROGRAM IS LOCATED
  878  000606  000000              MMAVA:  .WORD   0               ;MEMORY MANAGEMENT AVAILABLE FLAG.
```

```
                                        ;*****************************************************************
                                        .SBTTL  POWER DOWN AND UP ROUTINES
                                        ;POWER DOWN ROUTINE
000610  012737  000740  000024  $PWRDN: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
000616  012737  000340  000026         MOV     #340,@#PWRVEC+2  ;;PRIO:7
000624  010046                        MOV     R0,-(SP)         ;;PUSH R0 ON STACK
000626  010146                        MOV     R1,-(SP)         ;;PUSH R1 ON STACK
000630  010246                        MOV     R2,-(SP)         ;;PUSH R2 ON STACK
000632  010346                        MOV     R3,-(SP)         ;;PUSH R3 ON STACK
000634  010446                        MOV     R4,-(SP)         ;;PUSH R4 ON STACK
000636  010546                        MOV     R5,-(SP)         ;;PUSH R5 ON STACK
000640  010667  000100                MOV     SP,$SAVR6        ;;SAVE SP
000644  012737  000656  000024        MOV     #$PWRUP,@#PWRVEC ;;SET UP VECTOR
000652  000000                        HALT
000654  000776                        BR      .-2              ;;HANG UP
                                        ;POWER UP ROUTINE
000656  016706  000062         $PWRUP: MOV     $SAVR6,SP        ;;GET SP
000662  005067  000056                CLR     $SAVR6           ;;WAIT LOOP FOR THE TTY
000666  005267  000052         1$:    INC     $SAVR6           ;;WAIT FOR THE INC
000672  001375                        BNE     1$               ;;OF  WORD
000674  012605                        MOV     (SP)+,R5         ;;POP STACK INTO R5
000676  012604                        MOV     (SP)+,R4         ;;POP STACK INTO R4
000700  012603                        MOV     (SP)+,R3         ;;POP STACK INTO R3
000702  012602                        MOV     (SP)+,R2         ;;POP STACK INTO R2
000704  012601                        MOV     (SP)+,R1         ;;POP STACK INTO R1
000706  012600                        MOV     (SP)+,R0         ;;POP STACK INTO R0
000710  012737  000610  000024        MOV     #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
000716  012737  000340  000026        MOV     #340,@#PWRVEC+2  ;;PRIO:7
000724  004567  022310                JSR     R5,     $PRINT   ;GO PRINT OUT THE FOLLOWING MESSAGE.
000730  025363               $PWRMG: .WORD   PWRMSG           ;;POWER FAIL MESSAGE POINTER
000732  012716                        MOV     (PC)+,(SP)       ;;RESTART AT RESTART
000734  000300               $PWRAD: .WORD   RESTART          ;;RESTART ADDRESS
000736  000002                        RTI
000740  000000               $ILLUP: HALT                     ;;THE POWER UP SEQUENCE WAS STARTED
000742  000776                        BR      .-2              ;; BEFORE THE POWER DOWN WAS COMPLETE
000744  000000               $SAVR6: 0                        ;;PUT THE SP HERE
```

```
                                        ;*****************************************************************
                                        .SBTTL  COMMON TAGS
                                        ;*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
                                        ;*USED IN THE PROGRAM.
                                                .=1100
001100                          $CMTAG:                         ;;START OF COMMON TAGS
001100  000000                          .WORD   0
001102  000                     $TSTNM: .BYTE   0               ;;CONTAINS THE TEST NUMBER
001103  000                     $ERFLG: .BYTE   0               ;;CONTAINS ERROR FLAG
001104  000000                  $ICNT:  .WORD   0               ;;CONTAINS SUBTEST ITERATION COUNT
001106  000000                  $LPADR: .WORD   0               ;;CONTAINS SCOPE LOOP ADDRESS
001110  000000                  $LPERR: .WORD   0               ;;CONTAINS SCOPE RETURN FOR ERRORS
001112  000000                  $ERTTL: .WORD   0               ;;CONTAINS TOTAL ERRORS DETECTED
001114  000                     $ITEMB: .BYTE   0               ;;CONTAINS ITEM CONTROL BYTE
001115  001                     $ERMAX: .BYTE   1               ;;CONTAINS MAX. ERRORS PER TEST
001116  000000                  $ERRPC: .WORD   0               ;;CONTAINS PC OF LAST ERROR INSTRUCTION
001120  000000                  $GDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'GOOD' DATA
001122  000000                  $BDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'BAD' DATA
001124  000000                  $GDDAT: .WORD   0               ;;CONTAINS 'GOOD' DATA
001126  000000                  $BDDAT: .WORD   0               ;;CONTAINS 'BAD' DATA
001130  000000                          .WORD   0               ;;RESERVED--NOT TO BE USED
001132  000000                          .WORD   0
001134  000000                          .WORD   0
001136  177570                  SWR:    .WORD   DSWR            ;;ADDRESS OF SWITCH REGISTER
001140  177570                  DISPLAY: .WORD  DDISP           ;;ADDRESS OF DISPLAY REGISTER
001142  177560                  $TKS:   177560                  ;;TTY KBD STATUS
001144  177562                  $TKB:   177562                  ;;TTY KBD BUFFER
001146  177564                  $TPS:   177564                  ;;TTY PRINTER STATUS REG. ADDRESS
001150  177566                  $TPB:   177566                  ;;TTY PRINTER BUFFER REG. ADDRESS
001152  000                     $NULL:  .BYTE   0               ;;CONTAINS NULL CHARACTER FOR FILLS
001153  002                     $FILLS: .BYTE   2               ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
001154  012                     $FILLC: .BYTE   12              ;;INSERT FILL CHARS. AFTER A "LINE FEED"
001155  000                     $TPFLG: .BYTE   0               ;;"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
001156  000000                  $TMP0:  .WORD   0               ;;USER DEFINED
001160  000000                  $TMP1:  .WORD   0               ;;USER DEFINED
001162  000000                  $TMP2:  .WORD   0               ;;USER DEFINED
001164  000000                  $TMP3:  .WORD   0               ;;USER DEFINED
001166  000000                  $TIMES: 0                       ;;MAX. NUMBER OF ITERATIONS
001170  000000                  $ESCAPE:0                       ;;ESCAPE ON ERROR ADDRESS
001172  177607  000377           $BELL:  .ASCIZ  <207><377><377> ;;CODE FOR BELL
001176  077                     $QUES:  .ASCII  /?/             ;;QUESTION MARK
001177  015                     $CRLF:  .ASCII  <15>            ;;CARRIAGE RETURN
001200  000012                  $LF:    .ASCIZ  <12>            ;;LINE FEED
```

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

                                .SBTTL  APT MAILBOX-ETABLE


                                        .EVEN
001202                          $MAIL:                          ;;APT MAILBOX
001202  000000                  $MSGTY: .WORD   AMSGTY   ;;MESSAGE TYPE CODE
001204  000000                  $FATAL: .WORD   AFATAL   ;;FATAL ERROR NUMBER
001206  000000                  $TESTN: .WORD   ATESTN   ;;TEST NUMBER
001210  000000                  $PASS:  .WORD   APASS    ;;PASS COUNT
001212  000000                  $DEVCT: .WORD   ADEVCT   ;;DEVICE COUNT
001214  000000                  $UNIT:  .WORD   AUNIT    ;;I/O UNIT NUMBER
001216  000000                  $MSGAD: .WORD   AMSGAD   ;;MESSAGE ADDRESS
001220  000000                  $MSGLG: .WORD   AMSGLG   ;;MESSAGE LENGTH
001222                          $ETABLE:                        ;;APT ENVIRONMENT TABLE
001222      000                 $ENV:   .BYTE   AENV     ;;ENVIRONMENT BYTE
001223      000                 $ENVM:  .BYTE   AENVM    ;;ENVIRONMENT MODE BITS
001224  000000                  $SWREG: .WORD   ASWREG   ;;APT SWITCH REGISTER
001226  000000                  $USWR:  .WORD   AUSWR    ;;USER SWITCHES
001230  000000                  $CPUOP: .WORD   ACPUOP   ;;CPU TYPE,OPTIONS
                                ;*                              BITS 15-11=CPU TYPE
                                ;*                                      11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
                                ;*                                      11/70=06,PDQ=07,Q=10
                                ;*                              BIT 10=REAL TIME CLOCK
                                ;*                              BIT  9=FLOATING POINT PROCESSOR
                                ;*                              BIT  8=MEMORY MANAGEMENT
001232      000                 $MAMS1: .BYTE   AMAMS1   ;;HIGH ADDRESS,M.S. BYTE
001233      000                 $MTYP1: .BYTE   AMTYP1   ;;MEM. TYPE,BLK#1
                                ;*                              MEM.TYPE BYTE   --  (HIGH BYTE)
                                ;*                                      900 NSEC CORE=001
                                ;*                                      300 NSEC BIPOLAR=002
                                ;*                                      500 NSEC MOS=003
001234  000000                  $MADR1: .WORD   AMADR1   ;;HIGH ADDRESS,BLK#1
                                ;*                               MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
001236      000                 $MAMS2: .BYTE   AMAMS2   ;;HIGH ADDRESS,M.S. BYTE
001237      000                 $MTYP2: .BYTE   AMTYP2   ;;MEM.TYPE,BLK#2
001240  000000                  $MADR2: .WORD   AMADR2   ;;MEM.LAST ADDRESS,BLK#2
001242      000                 $MAMS3: .BYTE   AMAMS3   ;;HIGH ADDRESS,M.S.BYTE
001243      000                 $MTYP3: .BYTE   AMTYP3   ;;MEM.TYPE,BLK#3
001244  000000                  $MADR3: .WORD   AMADR3   ;;MEM.LAST ADDRESS,BLK#3
001246      000                 $MAMS4: .BYTE   AMAMS4   ;;HIGH ADDRESS,M.S.BYTE
001247      000                 $MTYP4: .BYTE   AMTYP4   ;;MEM.TYPE,BLK#4
001250  000000                  $MADR4: .WORD   AMADR4   ;;MEM.LAST ADDRESS,BLK#4
001252      000                 $VECT1: .BYTE   AVECT1   ;;INTERRUPT VECTOR#1
001253      000                 $VECT2: .BYTE   AVECT2   ;;INTERRUPT VECTOR#2
001254      000                 $PRIOR: .BYTE   APRIOR   ;;BUS PRIORITY #1,#2
001255      000                         .BYTE   0        ;;SPARE,NOT USED
                                        .EVEN
001256  000000                  $BASE:  .WORD   ABASE    ;;BASE ADDRESS OF EQUIPMENT UNDER TEST
001260  000000                  $DEVM:  .WORD   ADEVM    ;;DEVICE MAP
001262  000000                  $CDW1:  .WORD   ACDW1    ;;CONTROLLER DESCRIPTION WORD#1
001264  000000                  $CDW2:  .WORD   ACDW2    ;;CONTROLLER DESCRIPTION WORD#2
001266  000000                  $DDW0:  .WORD   ADDW0    ;;DEVICE DESCRIPTOR WORD#0
```

```
1020  001270  000000                $DDW1:  .WORD   ADDW1   ;;DEVICE DESCRIPTOR WORD#1
1021  001272  000000                $DDW2:  .WORD   ADDW2   ;;DEVICE DESCRIPTOR WORD#2
1022  001274  000000                $DDW3:  .WORD   ADDW3   ;;DEVICE DESCRIPTOR WORD#3
1023  001276  000000                $DDW4:  .WORD   ADDW4   ;;DEVICE DESCRIPTOR WORD#4
1024  001300  000000                $DDW5:  .WORD   ADDW5   ;;DEVICE DESCRIPTOR WORD#5
1025  001302  000000                $DDW6:  .WORD   ADDW6   ;;DEVICE DESCRIPTOR WORD#6
1026  001304  000000                $DDW7:  .WORD   ADDW7   ;;DEVICE DESCRIPTOR WORD#7
1027  001306  000000                $DDW8:  .WORD   ADDW8   ;;DEVICE DESCRIPTOR WORD#8
1028  001310  000000                $DDW9:  .WORD   ADDW9   ;;DEVICE DESCRIPTOR WORD#9
1029  001312  000000                $DDW10: .WORD   ADDW10  ;;DEVICE DESCRIPTOR WORD#10
1030  001314  000000                $DDW11: .WORD   ADDW11  ;;DEVICE DESCRIPTOR WORD#11
1031  001316  000000                $DDW12: .WORD   ADDW12  ;;DEVICE DESCRIPTOR WORD#12
1032  001320  000000                $DDW13: .WORD   ADDW13  ;;DEVICE DESCRIPTOR WORD#13
1033  001322  000000                $DDW14: .WORD   ADDW14  ;;DEVICE DESCRIPTOR WORD#14
1034  001324  000000                $DDW15: .WORD   ADDW15  ;;DEVICE DESCRIPTOR WORD#15
1035  001326                        $ETEND:
1036                                        .MEXIT
1037                                ;*****************************************************************
1038
1039                                .SBTTL  APT PARAMETER BLOCK
1040                                ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
1041                                ;*****************************************************************
1042          001326                        .$X=.   ;;SAVE CURRENT LOCATION
1043          000024                        .=24    ;;SET POWER FAIL TO POINT TO START OF PROGRAM
1044  000024  000200                        200     ;;FOR APT START UP
1045          000044                        .=44    ;;POINT TO APT INDIRECT ADDRESS PNTR.
1046  000044  001326                        $APTHDR ;;POINT TO APT HEADER BLOCK
1047          001326                        .=.$X   ;;RESET LOCATION COUNTER
1048                                ;*****************************************************************
1049                                ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
1050                                ;INTERFACE SPEC.
1051
1052  001326                        $APTHD:
1053  001326  000000                $HIBTS: .WORD   0       ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
1054  001330  001202                $MBADR: .WORD   $MAIL   ;;ADDRESS OF APT MAILBOX (BITS 0-15)
1055  001332  004540                $TSTM:  .WORD   2400.   ;;RUN TIM OF LONGEST TEST
1056  001334  000170                $PASTM: .WORD   120.    ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
1057  001336  000360                $UNITM: .WORD   240.    ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
1058  001340  000052                        .WORD   $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
1059                                ;*****************************************************************
1060
1061                                .SBTTL  APT STATISTICS TABLE
1062  001342                        $ASTAT:
1063  001342  177777  000000               .WORD   -1,0
1064  001346  177777  000000               .WORD   -1,0
1065  001352  177777  000000               .WORD   -1,0
1066  001356  177777  000000               .WORD   -1,0
1067  001362  177777  000000               .WORD   -1,0
1068  001366  177777  000000               .WORD   -1,0
1069  001372  177777  000000               .WORD   -1,0
1070  001376  177777  000000               .WORD   -1,0
1071  001402  177777  000000               .WORD   -1,0
1072  001406  177777  000000               .WORD   -1,0
1073  001412  177777  000000               .WORD   -1,0
1074  001416  177777  000000               .WORD   -1,0
1075  001422  177777  000000               .WORD   -1,0
```

```
1076  001426  177777  000000                  .WORD   -1,0
1077  001432  177777  000000                  .WORD   -1,0
1078  001436  177777  000000                  .WORD   -1,0
1079  001442  177777  000000                  .WORD   -1,0
1080  001446  177777  000000                  .WORD   -1,0
1081  001452  177777  000000                  .WORD   -1,0
1082  001456  177777  000000                  .WORD   -1,0
1083  001462  177777  000000                  .WORD   -1,0
1084  001466  177777  000000                  .WORD   -1,0
1085  001472  177777  000000                  .WORD   -1,0
1086  001476  177777  000000                  .WORD   -1,0
1087  001502  177777  000000                  .WORD   -1,0
1088  001506  177777                  $ASTEND:       -1
1089  001510  001342                  $APTR:  $ASTAT
1090
1091                                  ;*****************************************************************
1092                                  ;*THE FOLLOWING TAGS ARE USER DEFINED
1093                                  ;*****************************************************************
1094  001512  000000                  $VERPC: .WORD   0       ;VIRTUAL PC LOCATION FOR ERROR TYPEOUT ROUTINE ($ERTYP).
1095  001514  070032                  RESRVD: .WORD   070032  ;CORE PARITY REG BITS RESERVED FOR FUTURE USE.
1096                                                          ;NOTE:  FOR MS11 MEMORY WITH PARITY, CHANGE TO 077772.
1097  001516  000000                  LMAD:   .WORD   0       ;LAST CONTIGUOUS MEMORY ADDRESS (+2)
1098  001520  000000                  LDDISP: .WORD   0       ;CONTAINS DISPLAY REGISTER IMAGE
1099  001522                          MEMMAP:                 ;MEMORY MAP - EACH BIT CORRESPONDS TO 4K
1100  001522  000000                          .WORD   0       ;FIRST WORD CONTAINS LOW (0-64K) MAP
1101  001524  000000                          .WORD   0       ;SECOND WORD CONTAINS HIGH (64-128K) MAP
1102  001526                          TSTMAP:                 ;TEST MAP - WHICH BANKS ARE SELECTED FOR TEST.
1103  001526  000000                          .WORD   0       ;FIRST WORD CONTAINS LOW (0-64K) MAP
1104  001530  000000                          .WORD   0       ;SECOND WORD CONTAINS HIGH (64-128K) MAP
1105  001532                          SAVTST:                 ;SAVED TEST MAP - USED DURING FIRST PASS TO ONLY
1106                                                          ;  TEST EACH BANK ONCE.
1107  001532  000000                          .WORD   0       ;FIRST WORD CONTAINS LOW (0-64K) MAP
1108  001534  000000                          .WORD   0       ;SECOND WORD CONTAINS HIGH (64-128K) MAP
1109  001536                          PMEMAP:                 ;PARITY MAP - WHICH BANKS HAVE MEMORY PARITY
1110  001536  000000                          .WORD   0       ;FIRST WORD CONTAINS LOW (0-64K) MAP
1111  001540  000000                          .WORD   0       ;SECOND WORD CONTAINS HIGH (64-128K) MAP
1112  001542                          BITPT:                  ;POINTER TO CURRENT 4K BANK OF MEMORY
1113  001542  000000                          .WORD   0       ;FIRST WORD CONTAINS LOW (0-64K) MAP
1114  001544  000000                          .WORD   0       ;SECOND WORD CONTAINS HIGH (64-128K) MAP
1115  001546                          TMPPT:                  ;TEMPORARY POINTER FOR 2ND 4K BANK OF MEMORY
1116  001546  000000                          .WORD   0       ;FIRST WORD CONTAINS LOW (0-64K) MAP
1117  001550  000000                          .WORD   0       ;SECOND WORD CONTAINS HIGH (64-128K) MAP
1118  001552  000000                  MMORE:  .WORD   0       ;LOOP ADDRESS FOR MULTIPLE BLOCK TESTING.
1119                                                          ;SET UP BY "INITMM" AND "INITDN" ROUTINEES.
1120                                                          ;USED BY "MMUP" AND "MMDOWN" ROUTINES.
1121  001554  000                    SELFLG: .BYTE   0       ;OPERATOR SELECTED PARAMETERS FLAG.  (SA=204)
1122  001555  000                    FLAG8K: .BYTE   0       ;8K BLOCK INDICATOR.  USED IN "INITMM" AND "MMUP".
1123  001556  000                    OEFLG:  .BYTE   0       ;ODD/EVEN FLAG USED IN PARITY MEMORY BYTE TEST.
1124  001560                                  .EVEN
1125  001560  000000                  FSTADR: .WORD   0       ;FIRST VIRTUAL ADDRESS TO BE TESTED.
1126                                                          ;FIRST ADDRESS IS USER SELECTABLE.
1127  001562  000000                  TMPFAD: .WORD   0       ;ADJUSTED FIRST ADDRESS.
1128  001564  000000                  FADMSK: .WORD   0       ;BIT MASK TO ALLOW DOWNWARD ADDRESSING TESTS
1129                                                          ;  TO BREAK TO "MMDOWN" TO FIND FIRST ADDRESS.
1130  001566  000000  000000          FADMAP: .WORD   0,0     ;MAP OF BANK IN WHICH FIRST ADDRESS IS LOCATED.
1131  001572  000000                  LSTADR: .WORD   0       ;LAST VIRTUAL ADDRESS (+2) TO BE TESTED.
```

```
1132
1133  001574  000000                    TMPLAD: .WORD   0       ;LAST ADDRESS IS USER SELECTABLE.
1134  001576  000000                    LADMSK: .WORD   0       ;ADJUSTED LAST ADDRESS.
1135                                                            ;BIT MASK TO ALLOW UPWARD ADDRESSING TESTS
1136  001600  000000  000000                                    ; TO BREAK TO "MMUP" TO FIND LAST ADDRESS.
1137  001604  000000                    LADMAP: .WORD   0,0     ;MAP OF BANK IN WHICH LAST ADDRESS IS LOCATED.
1138  001606  000000                    BLKMSK: .WORD   0       ;BLOCK MASK, DETERMINES THE BLOCK SIZE.
1139                                    .CONST: .WORD   0       ;USER SELECTABLE CONSTANT DATA.
1140                                    ;*****************************************************************
1141                                    ;*  RELATIVE ADDRESSING TABLE.
1142                                    ;*  THE FOLLOWING LOCATIONS ARE MODIFIED AT RELOCATION TIME TO ALLOW
1143                                    ;*  RELATIVE ADDRESSING TO GET THE RELOCATED VALUE OF THE ARGUEMEMT TAGS.
1144                                    ;*****************************************************************
1145  001610                            RADTAB:
1146  001610  001100                    .STACK: STACK           ;STACK POINTER INITIAL ADDRESS.
1147  001612  001514                    .RESRV: RESRVD          ;PARITY REGISTER RESERVED BIT MASK ADDRESS.
1148  001614  002112                    .MPRO:  MPRO            ;MEMORY PARITY REGISTER TABLE ADDRESS.
1149  001616  002252                    .MPRX:  MPRX            ;MEMORY PARITY REGISTER EXIST TABLE ADDRESS.
1150  001620  014124                    .BRGOB: BRGOB           ;"BRANCH GOBBLE" ROUTINE ADDRESS.
1151  001622  014176                    .BGERR: BGERR           ;"BRANCH GOBBLE" ERROR ROUTINE ADDRESS.
1152  001624  014310                    .BGEXI: BGEXIT          ;"BRANCH GOBBLE" EXIT ROUTINE ADDRESS.
1153  001626  012456                    .PBTRP: PBTRP           ;PARITY BYTE TEST TRAP ROUTINE ADDRESS.
1154  001630  002064                    .MPPAT: MPPATS          ;MEMORY PARITY PATTERN TABLE ADDRESS.
1155  001632  017646                    .PESRV: PESRV           ;MEMORY PARITY ERROR TRAP ROUTINE ADDRESS.
1156  001634  002314                    .ERRTB: $ERRTB          ;ERROR TYPEOUT TABLE PONTER.
1157  001636  000010                    .EIGHT: 8.              ;DECIMAL TYPE ROUTINE COUNT DESIGNATOR.
1158  001640  014346                    .TST32: TST32           ;SCOPE ABORT ADR FOR WHEN NO MEM AVA FOR TEST.
1159                                    ;*****************************************************************
1160                                    ;*  DATA CONTAINERS FOR ERROR PRINTOUT.
1161                                    ;*****************************************************************
1162  001642  001116  001120  001124    DT1:    $ERRPC,$GDADR,$GDDAT,$BDDAT,0
1163  001650  001126  000000
1164  001654  001512  001116  001120    DT2:    $VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT,0
1165  001662  001124  001126  000000
1166  001670  001512  001116  001120    DT12:   $VERPC,$ERRPC,$GDADR,$GDDAT,0
1167  001676  001124  000000
1168  001702  001512  001116  001156    DT14:   $VERPC,$ERRPC,$TMP0,$GDADR,0
1169  001710  001120  000000
1170  001714  001512  001116  001120    DT15:   $VERPC,$ERRPC,$GDADR,$TMP0,$GDDAT,$BDDAT,0
1171  001722  001156  001124  001126
1172  001730  000000
1173  001732  001512  001116  001156    DT21:   $VERPC,$ERRPC,$TMP0,$GDADR,$GDDAT,$BDDAT,0
1174  001740  001120  001124  001126
1175  001746  000000
1176  001750  001156  001160  001162    DT22:   $TMP0,$TMP1,$TMP2,$TMP3,$GDADR,$GDDAT,$BDDAT,0
1177  001756  001164  001120  001124
1178  001764  001126  000000
1179  001770  001512  001116  001120    DT23:   $VERPC,$ERRPC,$GDADR,$BDADR,$GDDAT,$BDDAT,0
1180  001776  001122  001124  001126
1181  002004  000000
1182  002006  001512  001116  001122    DT24:   $VERPC,$ERRPC,$BDADR,0
1183  002014  000000
1184  002016  001512  001116  001122    DT25:   $VERPC,$ERRPC,$BDADR,$TMP0,$TMP1,0
1185  002024  001156  001160  000000
1186  002032  001512  001116  001156    DT26:   $VERPC,$ERRPC,$TMP0,$TMP1,0
1187  002040  001160  000000
```

```
1188  002044  001156  001160  001120  DT30:   $TMP0,$TMP1,$GDADR,$BDDAT,0
1189  002052  001126  000000
1190  002056  001164  000000          DT31:   $TMP3,0
1191  002062  177777                          .WORD   -1      ;TABLE TERMINATOR.
1192
1193                                  .SBTTL  MEMORY PARITY PATTERNS TABLE
1194                                  ;*****************************************************************
1195                                  ;THE FOLLOWING ARE THE PARITY PATTERNS EXERCISED THRUOUT MEMORY
1196                                  ;*****************************************************************
1197
1198  002064  125325                  MPPATS: 125325                  ;EVEN,ODD
1199  002066  152652                          152652                  ;ODD,EVEN
1200  002070  052452                          052452                  ;EVEN,ODD
1201  002072  025125                          025125                  ;ODD,EVEN
1202  002074  102070                          102070                  ;EVEN,EVEN
1203  002076  072527                          072527                  ;ODD,ODD
1204  002100  177777                          177777                  ;EVEN,EVEN
1205  002102  107030                          107030                  ;ODD,ODD
1206  002104  152525                          152525                  ;ODD,EVEN
1207  002106  000000                          0                       ;EXTRA PATTERN HOLDER FOR
1208                                                                  ;FUTURE USE
1209  002110  000000                  MPEND:  0                       ;TABLE TERMINATOR
1210
1211
1212                                  .SBTTL  MEMORY PARITY REGISTER ADDRESS TABLE
1213                                  ;///////////////////////////////////////////////////////////////
1214                                  ;*  THE FOLLOWING REPRESENTS THE MEMORY PARITY REGISTER ADDRESS TABLE
1215                                  ;*  FROM WHICH PARITY MEMORY IS ADDRESSED & CONTROLLED:
1216                                  ;*
1217                                  ;*  THE LEAST SIGNIFICANT BIT IN THE DEVICE ADDRESS IS SET TO A ONE (1)
1218                                  ;*  IF THE CONTROL IS FOUND NOT TO BE PRESENT.  THE MEMORY PRESENT UNDER
1219                                  ;*  THE CONTROL OF EACH CONTROLLER IS REPRESENTED BY TWO (2) WORDS FOLLOWING
1220                                  ;*  THE DEVICE ADDRESS,  EACH BIT REPRESENTING A 4K BLOCK.  I.E.
1221                                  ;*  FIRST WORD BIT0 = 0 - 4K,  BIT1 = 4 - 8K,...  BIT15 = 60 - 64K
1222                                  ;*  SECOND WORD BIT0 = 64 - 68K,... BIT14 = 120 - 124K.
1223                                  ;///////////////////////////////////////////////////////////////
1224  002112  172101                  MPR0:   172100 +1               ;PARITY STATUS REGISTER
1225  002114  000000                          0                       ;CONTROL MAP (LOW 64K)
1226  002116  000000                          0                       ;CONTROL MAP (HIGH 64K)
1227  002120  172103                  MPR1:   172102 +1               ;PARITY STATUS REGISTER
1228  002122  000000                          0                       ;CONTROL MAP (LOW 64K)
1229  002124  000000                          0                       ;CONTROL MAP (HIGH 64K)
1230  002126  172105                  MPR2:   172104 +1               ;PARITY STATUS REGISTER
1231  002130  000000                          0                       ;CONTROL MAP (LOW 64K)
1232  002132  000000                          0                       ;CONTROL MAP (HIGH 64K)
1233  002134  172107                  MPR3:   172106 +1               ;PARITY STATUS REGISTER
1234  002136  000000                          0                       ;CONTROL MAP (LOW 64K)
1235  002140  000000                          0                       ;CONTROL MAP (HIGH 64K)
1236  002142  172111                  MPR4:   172110 +1               ;PARITY STAUS REGISTER
1237  002144  000000                          0                       ;CONTROL MAP (LOW 64K)
1238  002146  000000                          0                       ;CONTROL MAP (HIGH 64K)
1239  002150  172113                  MPR5:   172112 +1               ;PARITY STATUS REGISTER
1240  002152  000000                          0                       ;CONTROL MAP (LOW 64K)
1241  002154  000000                          0                       ;CONTROL MAP (HIGH 64K)
1242  002156  172115                  MPR6:   172114 +1               ;PARITY STATUS REGISTER
1243  002160  000000                          0                       ;CONTROL MAP (LOW 64K)
```

```
1244   002162  000000                          0                       ;CONTROL MAP (HIGH 64K)
1245   002164  172117          MPR7:   172116 +1                       ;PARITY STATUS REGISTER
1246   002166  000000                  0                               ;CONTROL MAP (LOW 64K)
1247   002170  000000                  0                               ;CONTROL MAP (HIGH 64K)
1248   002172  172121          MPR8:   172120 +1                       ;PARITY STATUS REGISTER
1249   002174  000000                  0                               ;CONTROL MAP (LOW 64K)
1250   002176  000000                  0                               ;CONTROL MAP (HIGH 64K)
1251   002200  172123          MPR9:   172122 +1                       ;PARITY STATUS REGISTER
1252   002202  000000                  0                               ;CONTROL MAP (LOW 64K)
1253   002204  000000                  0                               ;CONTROL MAP (HIGH 64K)
1254   002206  172125          MPR10:  172124 +1                       ;PARITY STATUS REGISTER
1255   002210  000000                  0                               ;CONTROL MAP (LOW 64K)
1256   002212  000000                  0                               ;CONTROL MAP (HIGH 64K)
1257   002214  172127          MPR11:  172126 +1                       ;PARITY STATUS REGISTER
1258   002216  000000                  0                               ;CONTROL MAP (LOW 64K)
1259   002220  000000                  0                               ;CONTROL MAP (HIGH 64K)
1260   002222  172131          MPR12:  172130 +1                       ;PARITY STATUS REGISTER
1261   002224  000000                  0                               ;CONTROL MAP (LOW 64K)
1262   002226  000000                  0                               ;CONTROL MAP (HIGH 64K)
1263   002230  172133          MPR13:  172132 +1                       ;PARITY STATUS REGISTER
1264   002232  000000                  0                               ;CONTROL MAP (LOW 64K)
1265   002234  000000                  0                               ;CONTROL MAP (HIGH 64K)
1266   002236  172135          MPR14:  172134 +1                       ;PARITY STATUS REGISTER
1267   002240  000000                  0                               ;CONTROL MAP (LOW 64K)
1268   002242  000000                  0                               ;CONTROL MAP (HIGH 64K)
1269   002244  172137          MPR15:  172136 +1                       ;PARITY STATUS REGISTER
1270   002246  000000                  0                               ;CONTROL MAP (LOW 64K)
1271   002250  000000                  0                               ;CONTROL MAP (HIGH 64K)
1272                          ;THIS IS THE END OF THE TABLE !
1273   002252  000021          MPRX:   .BLKW   17.     ;TABLE TO HOLD JUST PARITY STATUS REGISTERS THAT EXIST.
1274                                                  ; (THE EXTRA WORD IS FOR A TERMINATOR.)
1275
1276
1277                          .SBTTL  ERROR POINTER TABLE
1278
1279                          ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
1280                          ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
1281                          ;*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
1282                          ;*NOTE1:        IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
1283                          ;*NOTE2:        EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
1284
1285                          ;*      EM              ;;POINTS TO THE ERROR MESSAGE
1286                          ;*      DH              ;;POINTS TO THE DATA HEADER
1287                          ;*      DT              ;;POINTS TO THE DATA
1288                          ;*      DF              ;;POINTS TO THE DATA FORMAT
1289
1290
1291   002314                 $ERRTB:
1292                          ;*  ITEM 1
1293   002314  026367                  DM1             ;PARITY REGISTER DATA ERROR.
1294   002316  027771                  DH1             ;PC,REG,S/B,WAS
1295   002320  001642                  DT1             ;$ERRPC,$GDADR,$GDDAT,$BDDAT
1296   002322  030401                  DF1             ;16,18,16,16
1297                          ;*  ITEM 2
1298   002324  026423                  DM2             ;ADDRESS TEST ERROR(TST1-5).
1299   002326  030010                  DH2             ;V/PC,P/PC,MA,S/B,WAS
```

```
1300  002330  001654                        DT2       ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
1301  002332  030405                        DF2       ;16,18,18,16,16
1302                          ;*  ITEM 3
1303  002334  026423                        DM2       ;ADDRESS TEST ERROR(TST1-5).
1304  002336  030010                        DH2       ;V/PC,P/PC,MA,S/B,WAS
1305  002340  001654                        DT2       ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
1306  002342  030412                        DF3       ;16,18,18,8,8
1307                          ;*  ITEM 4
1308  002344  026457                        DM4       ;CONSTANT DATA ERROR(TST6-10).
1309  002346  030010                        DH2       ;V/PC,P/PC,MA,S/B,WAS
1310  002350  001654                        DT2       ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
1311  002352  030405                        DF2       ;16,18,18,16,16
1312                          ;*  ITEM 5
1313  002354  026515                        DM5       ;ROTATING BIT ERROR(TST11-12).
1314  002356  030010                        DH2       ;V/PC,P/PC,MA,S/B,WAS
1315  002360  001654                        DT2       ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
1316  002362  030405                        DF2       ;16,18,18,16,16
1317                          ;*  ITEM 6
1318  002364  026553                        DM6       ;1 XOR 8 PATTERN ERROR (TST13).
1319  002366  030010                        DH2       ;V/PC,P/PC,MA,S/B,WAS
1320  002370  001654                        DT2       ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
1321  002372  030405                        DF2       ;16,18,18,16,16
1322                          ;*  ITEM 7
1323  002374  026611                        DM7       ;3 XOR 9 PATTERN ERROR(TST14-17).
1324  002376  030010                        DH2       ;V/PC,P/PC,MA,S/B,WAS
1325  002400  001654                        DT2       ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
1326  002402  030405                        DF2       ;16,18,18,16,16
1327                          ;*  ITEM 10
1328  002404  026652                        DM10      ;8 XOR 13 PATTERN ERROR(TST20).
1329  002406  030010                        DH2       ;V/PC,P/PC,MA,S/B,WAS
1330  002410  001654                        DT2       ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
1331  002412  030405                        DF2       ;16,18,18,16,16
1332                          ;*  ITEM 11
1333  002414  026711                        DM11      ;PARITY MEMORY ADDRESS ERROR(TST21).
1334  002416  030010                        DH2       ;V/PC,P/PC,MA,S/B,WAS
1335  002420  001654                        DT2       ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
1336  002422  030412                        DF3       ;16,18,18,8,8
1337                          ;*  ITEM 12
1338  002424  026755                        DM12      ;DATIP WITH WRONG PARITY DIDN'T TRAP(TST21).
1339  002426  030035                        DH12      ;V/PC,P/PC,MA,S/B
1340  002430  001670                        DT12      ;$VERPC,$ERRPC,$GDADR,$GDDAT
1341  002432  030412                        DF3       ;16,18,18,8
1342                          ;*  ITEM 13
1343  002434  027031                        DM13      ;WRONG PARITY TRAPED, BUT NO REGISTER SHOWS ERROR FLAG.
1344  002436  030035                        DH12      ;V/PC,P/PC,MA,S/B
1345  002440  001670                        DT12      ;$VERPC,$ERRPC,$GDADR,$GDDAT
1346  002442  030412                        DF3       ;16,18,18,8
1347                          ;*  ITEM 14
1348  002444  027121                        DM14      ;PARITY REGISTER NOT MAPPED AS CONTROLLING THIS ADDRESS(TST21).
1349  002446  030056                        DH14      ;V/PC,P/PC,REG,MA
1350  002450  001702                        DT14      ;$VERPC,$ERRPC,$TMPO,$GDADR
1351  002452  030417                        DF14      ;16,18,18,18
1352                          ;*  ITEM 15
1353  002454  026367                        DMI       ;PARITY REGISTER DATA ERROR.
1354  002456  030077                        DH15      ;V/PC,P/PC,MAUT,REG,S/B,WAS
1355  002460  001714                        DT15      ;$VERPC,$ERRPC,$GDADR,$TMPO,$GDDAT,$BDDAT
```

```
1356  002462  030417                          DF14        ;16,18,18,18,16,16
1357                              ;*  ITEM 16
1358  002464  027220                          DM16        ;MORE THAN ONE REGISTER INDICATED PARITY ERROR.
1359  002466  030056                          DH14        ;V/PC,P/PC,REG,MA
1360  002470  001702                          DT14        ;$VERPC,$ERRPC,$TMP0,$GDADR
1361  002472  030417                          DF14        ;16,18,18,18
1362                              ;*  ITEM 17
1363  002474  027277                          DM17        ;DATA SHOULDN'T HAVE CHANGED WHEN PARITY ERROR
1364                                                       ;  TRAPPED(TST21).
1365  002476  030010                          DH2         ;V/PC,P/PC,MA,S/B,WAS
1366  002500  001654                          DT2         ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
1367  002502  030412                          DF3         ;16,18,18,8,8
1368                              ;*  ITEM 20
1369  002504  027375                          DM20        ;RANDOM DATA ERROR(TST22).
1370  002506  030010                          DH2         ;V/PC,P/PC,MA,S/B,WAS
1371  002510  001654                          DT2         ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
1372  002512  030405                          DF2         ;16,18,18,16,16
1373                              ;*  ITEM 21
1374  002514  027427                          DM21        ;INSTRUCTION EXICUTION ERROR(TST23-30).
1375  002516  030132                          DH21        ;V/PC,P/PC,IUT,MA,S/B,WAS
1376  002520  001732                          DT21        ;$VERPC,$ERRPC,$TMP0,$GDADR,$GDDAT,$BDDAT
1377  002522  030425                          DF21        ;16,18,16,18,16,16
1378                              ;*  ITEM 22
1379  002524  027476                          DM22        ;"BRANCH GOBBLE" ERROR(TST31).
1380  002526  030163                          DH22        ;V/PC,P/PC,PS S/B,PS WAS,MA,S/B,WAS
1381  002530  001750                          DT22        ;$TMP0,$TMP1,$TMP2,$TMP3,$GDADR,$GDDAT,$BDDAT
1382  002532  030433                          DF22        ;16,18,16,16,18,16,16
1383                              ;*  ITEM 23
1384  002534  027534                          DM23        ;PROGRAM CODE CHANGED WHEN RELOCATED.
1385  002536  030226                          DH23        ;V/PC,P/PC,SRC MA,DST MA,S/B,WAS
1386  002540  001770                          DT23        ;$VERPC,$ERRPC,$GDADR,$BDADR,$GDDAT,$BDDAT
1387  002542  030417                          DF14        ;16,18,18,18,16,16
1388                              ;*  ITEM 24
1389  002544  027601                          DM24        ;TRAPPED, BUT NO REGISTER HAD ERROR BIT SET.
1390  002546  030266                          DH24        ;V/PC,P/PC,TRP/PC
1391  002550  002006                          DT24        ;$VERPC,$ERRPC,$BDADR
1392  002552  030417                          DF14        ;16,18,18
1393                              ;*  ITEM 25
1394  002554  027655                          DM25        ;TRAPPED TO 114.
1395  002556  030307                          DH25        ;V/PC,P/PC,TRP/PC,REG,WAS
1396  002560  002016                          DT25        ;$VERPC,$ERRPC,$BDADR,$TMP0,$TMP1
1397  002562  030417                          DF14        ;16,18,18,18,16
1398                              ;*  ITEM 26
1399  002564  027675                          DM26        ;FAILED TO TRAP.
1400  002566  030340                          DH26        ;V/PC,P/PC,REG,WAS
1401  002570  002032                          DT26        ;$VERPC,$ERRPC,$TMP0,$TMP1
1402  002572  030405                          DF2         ;16,18,18,16
1403                              ;*  ITEM 27
1404  002574  027715                          DM27        ;(ACTION ENABLE WASN'T SET).
1405  002576  030340                          DH26        ;V/PC,P/PC,REG,WAS
1406  002600  002032                          DT26        ;$VERPC,$ERRPC,$TMP0,$BDDAT
1407  002602  030405                          DF2         ;16,18,18,16
1408                              ;*  ITEM 30
1409  002604  000000                          0           ;NO MESSAGE.
1410  002606  030362                          DH30        ;REG,WAS,MA,WAS
1411  002610  002044                          DT30        ;$TMP0,$TMP1,$GDADR,$BDDAT
```

```
1412  002612  030442                          DF30        ;18,16,18,8
1413                                ;*  ITEM 31
1414  002614  027751                          DM31        ;TRAPPED TO 4
1415  002616  000000                          0           ;NO HEADER
1416  002620  002056                          DT31        ;$TMP3
1417  002622  030442                          DF30        ;18
```

```
1418                                    .SBTTL START:   SETUP AND MAP MEMORY

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

1425  002624  105067  176724          START:  CLRB    SELFLG          ;CLEAR SELECT PARAMETER FLAG.
1426  002630  000403                          BR      STARTA          ;GO DO SETUP AND MEMORY MAP.
1427  002632  112767  177777  176714  SELECT: MOVB    #-1,    SELFLG  ;SET THE SELECT PARAMETERS FLAG.
1428  002640                          STARTA:
1429  002640  012706  001100                  MOV     #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
1430  002644  005026                          CLR     (R6)+           ;;CLEAR MEMORY LOCATION
1431  002646  022706  001126                  CMP     #$BDDAT,R6      ;;DONE?
1432  002652  001374                          BNE     .-6             ;;LOOP BACK IF NO
1433  002654  012706  001100                  MOV     #STACK,SP       ;;SETUP THE STACK POINTER
1434  002660  012737  000610  000024          MOV     #$PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
1435  002666  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;LEVEL 7
1436  002674  013746  000004                  MOV     @#4,-(SP)       ;;SAVE ERROR VECTOR
1437  002700  013746  000006                  MOV     @#6,-(SP)
1438  002704  012767  002720  175072          MOV     #64$,4          ;;SET UP TIME OUT VECTOR
1439  002712  005777  176220                  TST     @SWR            ;;TRY TO REFERENCE HARDWARE SWR
1440  002716  000407                          BR      65$             ;;BRANCH IF NO TIMEOUT TRAP OCCURS
1441  002720  012767  000176  176210  64$:    MOV     #SWREG,SWR      ;;POINT TO SOFTWARE SWR
1442  002726  012767  000174  176204          MOV     #DISPREG,DISPLAY ;;POINT TO SOFTWARE DISPLAY REG
1443  002734  022626                          CMP     (SP)+,(SP)+     ;;RESTORE STACK
1444  002736  012637  000006          65$:    MOV     (SP)+,@#6       ;;RESTORE ERROR VECTOR
1445  002742  012637  000004                  MOV     (SP)+,@#4
1446  002746  005067  176236                  CLR     $PASS           ;;CLEAR PASS COUNT
1447  002752  132767  000200  176243          BITB    #APTSIZE,$ENVM  ;;TEST USER SIZE UNDER APT
1448  002760  001403                          BEQ     3$              ;;YES,USE NON-APT SWITCH
1449  002762  012767  001224  176146          MOV     #$SWREG,SWR     ;;NO,USE APT SWITCH REGISTER
1450  002770                          3$:
1451  002770  005067  176524                  CLR     LDDISP          ;CLEAR DISPLAY REGISTER STORAGE LOCN
1452  002774  005077  176140                  CLR     @DISPLAY        ;CLEAR DISPLAY REGISTER
1453  003000  005227  177777                  INC     #-1             ;;FIRST TIME?
1454  003004  001017                          BNE     66$             ;;BRANCH IF NO
1455  003006  004567  020226                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
1456  003012  003016                          .WORD   67$             ;ADDRESS OF MESSAGE TO BE TYPED
1457  003014  000413                          BR      66$             ;;GET OVER THE ASCIZ
1458                          ;;67$:  .ASCIZ  <200>'MAINDEC-11-DZQMC-B'<200>
1459  003044                  66$:
1460  003044  010700                          MOV     PC,     R0      ;GET CURRENT PROGRAM COUNTER.
1461  003046  022700  003046                  CMP     #.,     R0      ;CHECK IF THE PROGRAM IS RELOCATED.
1462  003052  001402                          BEQ     10$             ;BR IF PROGRAM NOT RELOCATED.
1463  003054  000167  175220                  JMP     RESTAR          ;GO TRY TO RELOCTED BEFORE CONTINUING.
1464  003060  012767  000003  175514  10$:    MOV     #3,     PRGMAP  ;INITIALIZE PROGRAM MAP....LO 64K.
1465  003066  005067  175512                  CLR     PRGMAP+2        ;...HI 64K.
1466  003072  005067  175502                  CLR     RELOCF          ;INIT THE RELOCATION FACTOR.
1467
1468                          ;*  ROUTINE TO SAVE THE LOADERS AT THE END OF 8K.
1469  003076  005767  176414                  TST     LMAD            ;CHECK IF LOADERS HAVE BEEN SAVED ALREADY.
1470  003102  001024                          BNE     14$             ;BRANCH IF ALREADY SAVED
1471  003104  012700  040000                  MOV     #40000, R0      ;GET END OF 8K
1472  003110  010001                          MOV     R0,     R1      ;GET END OF 8K
1473  003112  012737  003124  000004          MOV     #12$, @#ERRVEC  ;SET UP TIMEOUT VECTOR
```

```
 1474  003120  011020                  11$:    MOV     (R0),   (R0)+   ;SEARCH FOR END OF MEMORY
 1475  003122  000776                          BR      11$             ;KEEP SEARCHING
 1476  003124  022626                  12$:    CMP     (SP)+,  (SP)+   ;RESTORE STACK POINTER
 1477  003126  012737  024636  000004          MOV     #ERRTRP,@#ERRVEC ;RESET TIMEOUT VECTOR.
 1478  003134  010046                          MOV     R0,     -(SP)   ;SAVE LAST MEMORY ADDRESS (CONTIGUOUS)
 1479  003136  012702  002734                  MOV     #1500., R2      ;SET UP WORD COUNTER
 1480  003142  014041                  13$:    MOV     -(R0),  -(R1)   ;SAVE THE LOADERS
 1481  003144  005302                          DEC     R2              ;COUNT THE WORDS
 1482  003146  001375                          BNE     13$             ;BRANCH IF MORE WORDS
 1483  003150  012667  176342                  MOV     (SP)+,  LMAD    ;SAVE LAST MEMORY ADDRESS
 1484
 1485                                  ;*  CHECK IF MEMORY MANAGEMENT IS AVAILABLE, AND SET IT UP IF IT IS.
 1486  003154  005067  175426          14$:    CLR     MMAVA           ;CLEAR MEM MGMT AVAILABLE FLAG
 1487  003160  032777  010000  175750          BIT     #SW12,  @SWR    ;CHECK FOR INHIBIT KT11 SWITCH
 1488  003166  001014                          BNE     MAPMEM          ;BRANCH IF SET
 1489  003170  012737  003220  000004          MOV     #MAPMEM,@#ERRVEC ;SET UP TIMEOUT TRAP VECTOR
 1490  003176  005037  177572                  CLR     @#SR0           ;CLEAR MEM MGMT STATUS REG
 1491  003202  004767  011374                  JSR     PC,     MMINIT  ;MEM MGMT INITIALIZATION ROUTINE.
 1492  003206  005267  175374                  INC     MMAVA           ;SET MEM MGMT AVAILABLE FLAG
 1493  003212  004567  020022                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 1494  003216  025076                          .WORD   MMAMES          ;ADDRESS OF MESSAGE TO BE TYPED
 1495                                                                  ;"KT11 AVAILABLE"
 1496
 1497                                  ;*****************************************************************
 1498                                  ;*  ROUTINE TO MAP ALL OF MEMORY.
 1499                                  ;*  ONLY FULL 4K BANKS WILL BE RECOGNIZED.
 1500                                  ;*      R0 = MEMMAP POINTER...LO 64K.
 1501                                  ;*      R1 = MEMMAP POINTER...HI 64K.
 1502                                  ;*      R2 = ADDRESS POINTER
 1503                                  ;*      R3 = BANK POINTER...LO 64K.
 1504                                  ;*      R4 = BANK POINTER...HI 64K.
 1505                                  ;*      R5 = SCRATCH REGISTER.
 1506                                  ;*****************************************************************
 1507  003220  012706  001100          MAPMEM: MOV     #STACK, SP      ;RESET THE STACK
 1508  003224  012700  001522                  MOV     #MEMMAP,R0      ;SET UP MEMORY MAP POINTER...LO 64K.
 1509  003230  012701  001524                  MOV     #MEMMAP+2,R1    ;...HI 64K.
 1510  003234  005010                          CLR     (R0)            ;CLR MEMORY MAP...LO 64K.
 1511  003236  005011                          CLR     (R1)            ;...HI 64K.
 1512  003240  005002                          CLR     R2              ;SET ADDRESS POINTER TO 0
 1513  003242  012703  000001                  MOV     #1,     R3      ;SETUP 4K BANK POINTER...LO 64K.
 1514  003246  005004                          CLR     R4              ;...HI 64K.
 1515  003250  005067  175710                  CLR     $TMP3           ;INIT TEMPORARY HIGH ADDRESS BITS.
 1516  003254  004567  017760                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 1517  003260  025143                          .WORD   MEMMES          ;ADDRESS OF MESSAGE TO BE TYPED
 1518                                                                  ;"MEMORY MAP:"
 1519  003262  012737  003376  000004          MOV     #2$, @#ERRVEC   ;SET UP TIMEOUT VECTOR
 1520  003270  011222                  1$:     MOV     (R2),   (R2)+   ;READ+WRITE ALL MEMORY
 1521  003272  032702  017777                  BIT     #MASK4K,R2      ;CHECK FOR 4K BOUNDRY
 1522  003276  001374                          BNE     1$              ;BRANCH IF MORE IN BANK
 1523  003300  050310                          BIS     R3,     (R0)    ;SET FLAG FOR BANK...LO 64K.
 1524  003302  050411                          BIS     R4,     (R1)    ;...HI 64K.
 1525  003304  010267  175652                  MOV     R2,     $TMP2   ;SAVE ADDRESS POINTER.
 1526  003310  005367  175646                  DEC     $TMP2           ;ADJUST TO LAST ADR, LAST BANK.
 1527  003314  005767  175266                  TST     M1AVA           ;CHECK FOR MEM MGMT.
 1528  003320  001432                          BEQ     3$              ;BR IF NO MEM MGMT.
 1529  003322  042767  160000  175632          BIC     #160000,$TMP2   ;CLEAR BANK BITS ON RELATIVE ADDRESS.
```

```
 1530  003330  013705  172344                    MOV     @#KIPAR2,R5     ;SAVE KIPAR2.
 1531  003334  005067  175624                    CLR     $TMP3           ;MAKE SURE HI BITS ARE INIT.
 1532  003340  006305                            ASL     R5              ;SHIFT IT 6 PLACES.
 1533  003342  006305                            ASL     R5
 1534  003344  006305                            ASL     R5
 1535  003346  006305                            ASL     R5
 1536  003350  006305                            ASL     R5
 1537  003352  006167  175606                    ROL     $TMP3
 1538  003356  006305                            ASL     R5
 1539  003360  006167  175600                    ROL     $TMP3
 1540  003364  060567  175572                    ADD     R5,     $TMP2   ;MAKE LAST ADR PHYSICAL.
 1541  003370  005567  175570                    ADC     $TMP3
 1542  003374  000404                            BR      3$              ;GO TO UPDATE POINTERS.
 1543
 1544                                    ;*  TIMEOUT TRAPS TO HERE
 1545  003376  022626                    2$:     CMP     (SP)+,  (SP)+   ;RESTORE THE STACK POINTER
 1546  003400  052702  017777                    BIS     #MASK4K,R2      ;LAST ADDRESS OF 4K BANK
 1547  003404  005202                            INC     R2              ;FIRST ADDRESS OF NEXT BANK.
 1548  003406  005767  175174            3$:     TST     MMAVA           ;CHECK FOR MEM MGMT
 1549  003412  001411                            BEQ     4$              ;BRANCH IF NO MEM MGMT
 1550  003414  062737  000200  172344            ADD     #200, @#KIPAR2  ;UPDATE THIRD PAR
 1551  003422  012702  040000                    MOV     #40000, R2      ;POINT TO START OF THIRD PAR
 1552  003426  006303                            ASL     R3              ;UPDATE LO BANK POINTER.
 1553  003430  006104                            ROL     R4              ;UPDATE HI BANK POINTER
 1554  003432  100316                            BPL     1$              ;BRANCH IF MORE MEMORY TO MAP.
 1555  003434  000402                            BR      5$              ;EXIT WHEN DONE.
 1556
 1557  003436  106303                    4$:     ASLB    R3              ;UPDATE MAP POINTER
 1558  003440  100313                            BPL     1$              ;BRANCH IF NOT YET DONE
 1559  003442  012737  024636  000004    5$:     MOV     #ERRTRP, @#ERRVEC ;RESET TIMEOUT VECTOR
 1560  003450  004767  015154                    JSR     PC,     TYPMAP  ;GO TYPE THE MAP.
 1561  003454  004567  017560                    JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 1562  003460  001177                            .WORD   $CRLF           ;ADDRESS OF MESSAGE TO BE TYPED
 1563  003462  011067  176044                    MOV     (R0),   SAVTST  ;SET UP TEST MAP...LO 64K.
 1564  003466  011167  176042                    MOV     (R1),   SAVTST+2 ;...HI 64K.
 1565  003472  011000                            MOV     (R0),   R0      ;GET LOW MEM MAP
 1566  003474  042700  177760                    BIC     #177760, R0     ;MASK ALL BUT BOTTOM 4 BANKS
 1567  003500  020027  000017                    CMP     R0,     #17     ;CHECK THAT BOTTOM 16K IS ALL THERE!
 1568  003504  001530                            BEQ     GMPR            ;BRANCH IF BOTTOM 16K EXISTS
 1569  003506  004567  017526                    JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 1570  003512  025246                            .WORD   INSUFF          ;ADDRESS OF MESSAGE TO BE TYPED
 1571                                                                    ;"FIRST 16K OF MEMORY NOT ALL THERE!"
 1572  003514  000000                    6$:     HALT                    ;FATAL ERROR HALT...
 1573                                                                    ;MEMORY IS NOT CONFIGURED TO RUN THIS PROGRAM.
 1574                                    ;*****************************************************************
 1575                                    ;*  SPECIAL ROUTINE TO TYPE OUT ALL UNIBUS ADDRESSES WHICH RESPOND TO
 1576                                    ;*   DATI, DATIP, DATO, AND DATOB.
 1577                                    ;*****************************************************************
 1578  003516  012706  001100    TIMOUT: MOV     #STACK, SP      ;SET UP THE STACK POINTER.
 1579  003522  005067  175060            CLR     MMAVA           ;CLEAR MEM MGMT AVAILABLE FLAG.
 1580  003526  032777  010000  175402    BIT     #SW12,  @SWR    ;CHECK IF MEM MGMT TO BE INHIBITED.
 1581  003534  001011                    BNE     1$              ;BR IF NO MEM MGMT.
 1582  003536  012737  003560  000004    MOV     #1$,    @#ERRVEC ;SET TIMEOUT FOR MEM MGMT CHECK.
 1583  003544  005037  177572            CLR     @#SR0           ;CHECK FOR MEM MGMT...TIMES OUT IF NONE.
 1584  003550  004767  011026            JSR     PC,     MMINIT  ;INIT ALL MEM MGMT REGISTERS.
 1585  003554  005267  175026            INC     MMAVA           ;SET MEM MGMT AVAILABLE FLAG.
```

```
1586  003560                          1$:
1587  003560  004567  017454                JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
1588  003564  025161                        .WORD   BYTMES          ;ADDRESS OF MESSAGE TO BE TYPED
1589                                                                ;"BYTE MEMORY MAP:"
1590  003566  005000                        CLR     R0              ;SET UP TYPE OUT FLAG.
1591  003570  005002                        CLR     R2              ;SET ADDRESS POINTER TO ZERO.
1592  003572  012737  003636  000004        MOV     #20$,   @#ERRVEC ;SET TIME OUT VEC TO SERVICE NON-EX MEM.
1593  003600  105712                10$:    TSTB    (R2)            ;DO DATI ONLY.
1594  003602  032702  000001                BIT     #BIT0,  R2      ;CHECK FOR WORD ADDRESS.
1595  003606  001001                        BNE     11$             ;BR IF ODD BYTE ADDRESS.
1596  003610  011212                        MOV     (R2),   (R2)    ;DO DATI, DATO...NOP FOR READ ONLY MAP.
1597  003612  151212                11$:    BISB    (R2),   (R2)    ;DO DATI, DATIP, DATOB... NOP FOR READ ONLY MAP.
1598  003614  005700                        TST     R0              ;CHECK FOR PREVIOUS TYPOUT.
1599  003616  001023                        BNE     30$             ;BR IF ALREADY TYPED "FROM".
1600  003620  004567  017414                JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
1601  003624  025231                        .WORD   FROM            ;ADDRESS OF MESSAGE TO BE TYPED
1602                                                                ;"FROM"
1603  003626  010246                        MOV     R2,     -(SP)   ;PUT THE DATA ON THE STACK.
1604  003630  004767  021040                JSR     PC,     $TYPAD  ;DETERMINE THE PHYSICAL ADDRESS AND TYPE IT.
1605  003634  000413                        BR      29$             ;GO TO ADDRESS POINTER UPDATE.
1606                                ;*  TIME OUTS COME HERE.
1607  003636  022626                20$:    CMP     (SP)+,  (SP)+   ;POP TWO OFF STACK.
1608  003640  005700                        TST     R0              ;CHECK FOR PREVIOUS TYPOUT.
1609  003642  001411                        BEQ     30$             ;BR IF ALREADY TYPED "TO".
1610  003644  004567  017370                JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
1611  003650  025241                        .WORD   TO              ;ADDRESS OF MESSAGE TO BE TYPED
1612                                                                ;"TO"
1613  003652  005302                        DEC     R2              ;BACK UP ONE BYTE.
1614  003654  010246                        MOV     R2,     -(SP)   ;PUT THE DATA ON THE STACK.
1615  003656  004767  021012                JSR     PC,     $TYPAD  ;DETERMINE THE PHYSICAL ADDRESS AND TYPE IT.
1616  003662  005202                        INC     R2              ;RESET ADDRESS POINTER.
1617  003664  005100                29$:    COM     R0              ;RESET PREVIOUS TYPOUT FLAG.
1618  003666  005202                30$:    INC     R2              ;UPDATE ADDRESS POINTER TO NEXT BYTE.
1619  003670  001423                        BEQ     31$             ;EXIT IF ZERO REACHED.
1620  003672  032702  017777                BIT     #MASK4K,R2      ;CHECK FOR 4K BANK BOUNDRY.
1621  003676  001340                        BNE     10$             ;BR IF MORE THIS 4K BANK.
1622  003700  005767  174702                TST     MMAVA           ;CHECK IF MEM MGMT IS AVAILABLE.
1623  003704  001735                        BEQ     10$             ;BR IF NO MEM MGMT.
1624  003706  022737  007600  172346        CMP     #7600,  @#KIPAR3 ;CHECK FOR END OF LAST 4K BANK.
1625  003714  001411                        BEQ     31$             ;EXIT WHEN ALL DONE.
1626  003716  012702  060000                MOV     #60000, R2      ;RESET VIRTUAL ADDRESS POINTER.
1627  003722  013737  172346  172344        MOV     @#KIPAR3,@#KIPAR2 ;SAVE MEM MGMT REG FOR TYPEOUT.
1628  003730  062737  000200  172346        ADD     #200,   @#KIPAR3 ;UPDATE MEM MGMT REG 2 TO NEXT 4K BANK.
1629  003736  000720                        BR      10$             ;BR BACK TO DO NEXT BANK.
1630  003740  005700                31$:    TST     R0              ;CHECK PREVIOUS TYPE FLAG BEFORE EXIT.
1631  003742  001407                        BEQ     32$             ;BR TO EXIT IF TYPING ALL DONE.
1632  003744  004567  017270                JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
1633  003750  025241                        .WORD   TO              ;ADDRESS OF MESSAGE TO BE TYPED
1634                                                                ;"TO"
1635  003752  005302                        DEC     R2              ;BACK ADDRESS POINTER UP ONE BYTE.
1636  003754  010246                        MOV     R2,     -(SP)   ;PUT THE DATA ON THE STACK.
1637  003756  004767  020712                JSR     PC,     $TYPAD  ;DETERMINE THE PHYSICAL ADDRESS AND TYPE IT.
1638  003762  000000                32$:    HALT                    ;* THIS ROUTINE IS FOR DEBUG USE ONLY.
1639                                                                ;*  TO RUN THE MAIN PROGRAM RESTART AT 200 OR 204.
1640  003764  000654                        BR      TIMOUT          ;LOOP BACK AND DO AGAIN UPON CONTINUE.
1641
```

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

003766  012704  002252          GMPR:   MOV     #MPRX,  R4      ;SET UP POINTER TO PARITY REG EXIST TABLE.
003772  032777  000100  175136          BIT     #SW06,  @SWR    ;CHECK FOR INHIBIT PARITY SWITCH.
004000  001035                        BNE     GMPRD           ;BR IF INHIBIT PARITY.
004002  012703  002112                MOV     #MPRD,  R3      ;SET UP TABLE POINTER
004006  012737  004030  000004        MOV     #GMPRB, @#ERRVEC ;SET UP TIMEOUT TRAP SERVICE
004014  042713  000001          GMPRA:  BIC     #1,     (R3)    ;CLEAR FLAG BIT IN TABLE
004020  005773  000000                TST     @(R3)           ;DOES THIS MEMORY PARITY REGISTER EXIST.
                                        ;*  IF IT DOESN'T EXIST, A TIMEOUT TRAP WILL GO TO "GMPRB".
004024  012324                        MOV     (R3)+,  (R4)+   ;SAVE IT IN THE PARITY REG EXIST TABLE.
004026  000403                        BR      GMPRC           ;SKIP TIMEOUT SERVICE CODE
                                        ;*  TIMEOUT COMES HERE
004030  022626                GMPRB:  CMP     (SP)+,  (SP)+   ;RESTORE STACK POINTER
004032  052723  000001                BIS     #1,     (R3)+   ;SET FLAG TO INDICATE REGISTER NOT PRESENT
004036  005023                GMPRC:  CLR     (R3)+           ;CLEAR THE MAP...LO 64K.
004040  005023                        CLR     (R3)+           ;...HI 64K.
004042  020327  002252                CMP     R3,     #MPRX   ;HAVE WE CHECKED ALL REGISTERS?
004046  103762                        BLO     GMPRA           ;NO - GO BACK TO CHECK NEXT ONE
004050  005014                        CLR     (R4)            ;SET TERMINATOR IN PARITY REG EXIST TABLE.
004052  012737  024636  000004        MOV     #ERRTRP,@#ERRVEC ; RESTORE TRAPCATCHER
004060  005767  176166                TST     MPRX                    ;ANY PARITY REGISTERS PRESENT?
004064  001006                        BNE     CTRLS                   ;YES - GO TEST CONTROLS PRESENT
004066  004567  017146                JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
004072  025327                        .WORD   MTR             ;ADDRESS OF MESSAGE TO BE TYPED
                                                                ;"NO MEMORY PARITY REGISTERS FOUND"
004074  005014                GMPRD:  CLR     (R4)            ;MAKE SURE TABLE IS CLEAR.
004076  000167  000544                JMP     MANUAL                  ;AND SKIP ALL CONTROLS TESTING

                                        .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.
                                        ;*************************************************************

004102  012703  002252          CTRLS:  MOV     #MPRX,  R3      ;LOAD INITIAL TABLE ADDRESS FOR A POINTER
004106  012302                  1$:     MOV     (R3)+,  R2      ;LOAD R2 WITH ADDRESS OF THIS PARITY REGISTER
004110  012700  000001                MOV     #1,     R0      ;LOAD R0 WITH VALUE OF 1ST BIT TESTED
004114  005012                        CLR     (R2)            ;INITIALIZE THE PARITY REGISTER
004116  011201                        MOV     (R2),   R1      ;READ THE CONTENTS OF THE PARITY REGISTER
004120  046701  175370                BIC     RESRVD, R1      ;CLEAR BITS WHICH ARE RESERVED
004124  001405                        BEQ     2$              ;CHECK OTHER BITS - BRANCH IF OK
004126  004767  014376          64$:  JSR     PC,     $PRNT   ;SET UP VALUES FOR ERROR PRINTING.
004132  004767  015700                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
004136  000001                        .WORD   1               ;ERROR TYPE CODE.
004140  030067  175350          2$:   BIT     R0,     RESRVD  ;IS THIS BIT RESERVED?
004144  001025                        BNE     3$              ;YES - DON'T TEST IT
004146  010012                        MOV     R0,     (R2)    ;NO - SET THIS BIT IN THE PARITY REGISTER
004150  011201                        MOV     (R2),   R1      ;READ & SAVE CONTENTS OF THE  PARITY REGISTER
004152  005012                        CLR     (R2)            ;CLEAR THE PARITY REGISTER
004154  046701  175334                BIC     RESRVD, R1      ;CLEAR BIT LOCATIONS THAT ARE RESERVED
```

```
1698                                    ;IMPORTANT NOTE:        THE FOLLOWING TEST WILL FAIL ON A PDP-11/45 WITH
1699                                    ;                       MOS PARITY REGISTERS.  PATCH "RESRVD"(1514) TO 77772.
1700  004160  020001                            CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
1701  004162  001405                            BEQ     66$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
1702  004164  004767  014370          65$:      JSR     PC,     SPRNTO  ;SET UP VALUES FOR ERROR PRINTING.
1703  004170  004767  015642                    JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
1704  004174  000001                            .WORD   1               ;ERROR TYPE CODE.
1705  004176                          66$:
1706  004176  011201                            MOV     (R2),   R1      ;READ THE CONTENTS OF THE PARITY REGISTER
1707  004200  046701  175310                    BIC     RESRVD, R1      ;CLEAR BITS WHICH ARE RESERVED
1708  004204  001405                            BEQ     3$              ;CHECK OTHER BITS - BRANCH IF OK
1709  004206  004767  014316          67$:      JSR     PC,     SPRNT   ;SET UP VALUES FOR ERROR PRINTING.
1710  004212  004767  015620                    JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
1711  004216  000001                            .WORD   1               ;ERROR TYPE CODE.
1712  004220  006300                    3$:     ASL     R0              ;ROTATE TO GET NEXT BIT TO BE TESTED
1713  004222  103346                            BCC     2$              ;BRANCH IF NOT DONE WITH ALL BITS
1714  004224  005713                            TST     (R3)            ;CHECK FOR TABLE TERMINATOR.
1715  004226  001327                            BNE     1$              ;BR IF MORE PARITY REGISTERS.
1716
1717                                    ;*****************************************************************
1718                                    ;* SHOW THAT RESET CLEARS BITS 0,2, AND 15 OF EACH PARITY REGISTER PRESENT.
1719                                    ;*  THIS IS A ONCE ONLY TEST.
1720                                    ;*****************************************************************
1721
1722  004230  012703  002252    RESCHK: MOV     #MPRX,  R3              ;LOAD INITIAL TABLE ADDRESS FOR A POINTER
1723  004234  012733  107745    1$:     MOV     #107745,@(R3)+          ;SET ALL DEFINED BITS TO A ONE (1)
1724  004240  005713                    TST     (R3)                    ;CHECK FOR TABLE TERMINATOR.
1725  004242  001374                    BNE     1$                      ;BRANCH IF MORE REGISTERS.
1726  004244  000005                    RESET                           ;ISSUE AN INIT TO THE WORLD.
1727  004246  012703  002252            MOV     #MPRX,  R3              ;LOAD INITIAL TABLE ADDRESS FOR A POINTER
1728  004252  012302            2$:     MOV     (R3)+,  R2              ;STORE THE PARITY REGISTER ADDRESS
1729  004254  011201                    MOV     (R2),   R1              ;GET CONTENTS OF REGISTER
1730  004256  005012                    CLR     (R2)                    ;MAKE SURE IT IS CLEAR.
1731  004260  042701  077772            BIC     #77772, R1              ;MASK THE BITS RESERVED FOR FOR FUTURE USE
1732  004264  005701                    TST     R1                      ;CHECK IF REST WERE CLEARED BY THE RESET
1733  004266  001405                    BEQ     65$                     ;BRANCH OVER ERROR CALL IF GOOD DATA.
1734  004270  004767  014234    64$:    JSR     PC,     SPRNT           ;SET UP VALUES FOR ERROR PRINTING.
1735  004274  004767  015536            JSR     PC,     $ERROR          ;*** ERROR *** (GO TYPE A MESSAGE)
1736  004300  000001                    .WORD   1                       ;ERROR TYPE CODE.
1737  004302                    65$:
1738  004302  005713                    TST     (R3)                    ;CHECK FOR TABLE TERMINATOR.
1739  004304  001362                    BNE     2$                      ;BR IF MORE PARITY REGISTERS.
```

```
                                .SBTTL  MAP PARITY MEMORY
                                ;*****************************************************************
                                ;MAP CORRESPONDENCE BETWEEN PARITY REGISTERS AND MEMORY, AND TYPE RESULTS
                                ;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
                                ;*****************************************************************

004306  004767  014174          JSR     PC,     CLRPAR  ;INITIALIZE ALL PARITY REGISTERS
004312  012767  000001  175222  MOV     #1,     BITPT   ;INITIALIZE 4K POINTER
004320  005067  175220          CLR     BITPT+2         ;CLEAR HI 64K POINTER
004324  005002                  CLR     R2              ;SET ADR POINTER TO ZERO
004326  005767  174254          TST     MMAVA           ;CHECK FOR MEM MGMT
004332  001404                  BEQ     MAPRB           ;BRANCH IF NO MEM MGMT
004334  012702  040000          MOV     #40000, R2      ;SET ADR POINTER TO PAR2
004340  004767  010236          JSR     PC,     MMINIT  ;SET UP ALL MEMORY MGMT REGISTERS.

                                ;*****************************************************************
                                ;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.
                                ;*****************************************************************

004344  005067  175166  MAPRB:  CLR     PMEMAP          ;CLEAR THE PARITY MEMORY MAP
004350  005067  175164          CLR     PMEMAP+2
004354  012703  002112  1$:     MOV     #MPRO,  R3      ;INITIALIZE TABLE ADDRESS
004360  032713  000001  2$:     BIT     #1,     (R3)    ;IS THIS REGISTER PRESENT?
004364  001027                  BNE     3$              ;NO - GET THE NEXT ONE
004366  012773  000004  000000  MOV     #WWP,   @(R3)   ;YES - SET WRITE WRONG PARITY
                                                        ;AND CLEAR REST OF REGISTER
004374  011212                  MOV     (R2),   (R2)    ;WRITE WRONG PARITY
004376  005712                  TST     (R2)            ;READ WRONG PARITY
004400  042773  000004  000000  BIC     #WWP,   @(R3)   ;CLEAR WRITE WRONG PARITY
004406  005773  000000          TST     @(R3)           ;OTHERWISE, CHECK TO SEE IF THIS
                                                        ;CONTROL REGISTER GOT A PARITY
                                                        ;ERROR
004412  100014                  BPL     3$              ;BRANCH IF IT DIDN'T AND CHECK
                                                        ;NEXT POSSIBLE ONE
004414  056763  175122  000002  BIS     BITPT,  2(R3)   ;SET FLAG IN MAP FOR THIS PARITY REGISTER
004422  056763  175116  000004  BIS     BITPT+2,4(R3)
004430  056767  175106  175100  BIS     BITPT,  PMEMAP  ;SET FLAG IN PARITY MAP
004436  056767  175102  175074  BIS     BITPT+2,PMEMAP+2
004444  062703  000006  3$:     ADD     #6.     R3      ;STEP UP TO NEXT REGISTER
004450  020327  002252          CMP     R3,     #MPRX   ;ARE WE DONE WITH TABLE?
004454  103741                  BLO     2$              ;GO BACK TO CHECK FOR ANY MORE!
004456  011212                  MOV     (R2),   (R2)    ;CLEAR BAD PARITY
004460  005767  174122          TST     MMAVA           ;CHECK FOR MEM MGMT
004464  001425                  BEQ     10$             ;BR IF NO MEM MGMT
004466  062737  000200  172344  4$:     ADD     #200,   @#KIPAR2 ;UPDATE PAR TO NEXT 4K BANK.
004474  006367  175042          ASL     BITPT           ;UPDATE BANK POINTER...LO 64K.
004500  006167  175040          ROL     BITPT+2         ;...HI 64K.
004504  100422                  BMI     TMAP            ;BR IF ALL DONE.
```

```
 1796  004506  036767  175030  175006          BIT     BITPT,  MEMMAP  ;CHECK IF BANK EXISTS...LO 64K.
 1797  004514  001317                          BNE     1$              ;BR IF BANK EXISTS.
 1798  004516  036767  175022  175000          BIT     BITPT+2,MEMMAP+2;...HI 64K.
 1799  004524  001313                          BNE     1$              ;BR IF BANK EXISTS.
 1800  004526  000757                          BR      4$              ;BR IF BANK DOESN'T EXIST.
 1801  004530  036767  175006  174764  11$:    BIT     BITPT,  MEMMAP  ;CHECK IF BANK EXISTS.
 1802  004536  001306                          BNE     1$              ;BR IF BANK EXISTS.
 1803  004540  062702  020000          10$:    ADD     #20000, R2      ;UPDATE ADDRESS POINTER TO NEXT BANK.
 1804  004544  106367  174772                  ASLB    BITPT           ;MOVE POINTER TO NEXT BANK.
 1805  004550  100367                          BPL     11$             ;BR IF MORE TO LOOK FOR.
 1806
 1807                                  ;*****************************************************************
 1808                                  ;*  ROUTINE TO TYPE MAP OF WHERE PARITY MEMORY IS PRESENT
 1809                                  ;*  AND WHICH CONTROL REGISTERS CONTROL WHICH MEMORY
 1810                                  ;*****************************************************************
 1811
 1812  004552  004767  013730          TMAP:   JSR     PC,     CLRPAR  ;INITIALIZE ALL PARITY REGISTERS PRESENT
 1813  004556  004567  016456                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 1814  004562  025204                          .WORD   MTMAP           ;ADDRESS OF MESSAGE TO BE TYPED
 1815                                                                  ;"PARITY MEMORY MAP:"
 1816  004564  012703  002112                  MOV     #MPR0,  R3      ;INITIALIZE TABLE POINTER
 1817  004570  032713  000001          1$:     BIT     #BIT0,  (R3)    ;CHECK IF THIS REGISTER IS PRESENT.
 1818  004574  001017                          BNE     2$              ;BR IF NOT PRESENT.
 1819  004576  004567  016436                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 1820  004602  025613                          .WORD   MX1             ;ADDRESS OF MESSAGE TO BE TYPED
 1821                                                                  ;"REGISTER AT"
 1822  004604  011346                          MOV     (R3),-(SP)      ;;SAVE (R3) FOR TYPEOUT
 1823                                  ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOC ROUTINE
 1824                                  ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
 1825  004606  013746  177776                  MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
 1826  004612  004767  017614                  JSR     PC,     $TYPOC  ;GO TO THE SUBROUTINE
 1827  004616  004567  016416                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 1828  004622  025632                          .WORD   MX2             ;ADDRESS OF MESSAGE TO BE TYPED
 1829                                                                  ;"CONTROLS"
 1830  004624  010300                          MOV     R3,     R0      ;SET UP R0 FOR TYPMAP ROUTINE.
 1831  004626  005720                          TST     (R0)+           ;UPDATE POINTER TO MAP.
 1832  004630  004767  013774                  JSR     PC,     TYPMAP  ;GO TYPE THE MEMORY COVERED BY THIS REGISTER.
 1833  004634  062703  000006          2$:     ADD     #6,     R3      ;UPDATE TO NEXT REGISTER IN TABLE.
 1834  004640  020327  002252                  CMP     R3,     #MPRX   ;ARE WE ALL DONE WITH TABLE?
 1835  004644  103751                          BLO     1$              ;BRANCH IF MORE REGISTERS
 1836
 1837  004646  012700  000014          MANUAL: MOV     #12.,   R0      ;SET COUNTER TO CLEAR 12 WORDS.
 1838  004652  012701  001560                  MOV     #FSTADR,R1      ;STARTING AT FSTADR.
 1839  004656  005021                  1$:     CLR     (R1)+           ;CLEAR THE LOCATIONS.
 1840  004660  005300                          DEC     R0              ;COUNT.
 1841  004662  001375                          BNE     1$              ;BR IF MORE.
 1842  004664  105767  174664                  TSTB    SELFLG          ;CHECK FOR SELECT PARAMETERS STARTUP.
 1843  004670  001005                          BNE     MANUL1          ;BR IF PARAMETERS TO BE SELECTED.
 1844  004672  016767  174264  174672          MOV     $TMP2,  LSTADR  ;SET UP VIRTUAL LAST ADDRESS.
 1845  004700  000167  000402                  JMP     MANUL2          ;SKIP PARAMETER SELECTION SECTION.
```

```
                                .SBTTL  USER PARAMETER SELECTION SECTION
                                ;*****************************************************************
                                ;* USER PARAMETER SELECTION SECTION IS ENTERED BY STARTING AT 204.
                                ;*****************************************************************
004704  012700  000001  MANUL1: MOV     #BIT0,  R0          ;SET UP BANK POINTER.
004710  005001                  CLR     R1                  ;...HI 64K.
004712  005002                  CLR     R2                  ;CLEAR ADDRESS POINTER.
004714  005003                  CLR     R3                  ;...HI ADDRESS BITS.
004716  004567  016316          JSR     R5,     $PRINT      ;GO PRINT OUT THE FOLLOWING MESSAGE.
004722  025670                  .WORD   FADMES              ;ADDRESS OF MESSAGE TO BE TYPED
                                                            ;"FIRST ADDRESS:"
                                ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $RDOCT ROUTINE
                                ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
004724  013746  177776          MOV     @#PSW,  -(SP)       ;PUT THE PROCESSOR STATUS ON THE STACK
004730  004767  016132          JSR     PC,     $RDOCT      ;GO TO THE SUBROUTINE
004734  042716  000001          BIC     #BIT0,  (SP)        ;MAKE SURE ADDRESS IS ON A WORD BOUNDRY.
004740  005067  174566          CLR     SAVTST              ;INIT TEST MAP...LO 64K.
004744  005067  174564          CLR     SAVTST+2            ;...HI 64K.
004750  062702  020000  1$:     ADD     #20000, R2          ;UPDATE ADDRESS POINTER TO NEXT BANK.
004754  005503                  ADC     R3
004756  020367  016254          CMP     R3,     $HIOCT      ;CHECK HI ADDRESS BITS.
004762  103403                  BLO     2$                  ;BR IF NOT HI ENOUGH YET.
004764  101006                  BHI     3$                  ;BR IF PAST SELECTED ADDRESS.
004766  020216                  CMP     R2,     (SP)        ;CHECK THE LO ADDRESS BITS.
004770  101004                  BHI     3$                  ;BR IF PAST SELECTED ADDRESS.
004772  006300  2$:     ASL     R0                          ;UPDATE POINTER...LO 64K.
004774  006101                  ROL     R1                  ;...HI 64K.
004776  100364                  BPL     1$                  ;BR BACK TO CHECK NEXT BANK.
005000  000507                  BR      17$                 ;BR IF OVERFLOW.
005002  030067  174514  3$:     BIT     R0,     MEMMAP      ;CHECK IF BANK EXISTS.
005006  001003                  BNE     4$                  ;BR IF BANK EXISTS.
005010  030167  174510          BIT     R1,     MEMMAP+2    ;CHECK HI 64K.
005014  001501                  BEQ     17$                 ;BR IF ADDRESS IN UN-MAPPED BANK.
005016  016704  016214  4$:     MOV     $HIOCT, R4          ;SAVE FIRST ADR HI BITS.
005022          10$:
005022  004567  016212          JSR     R5,     $PRINT      ;GO PRINT OUT THE FOLLOWING MESSAGE.
005026  025755                  .WORD   LADMES              ;ADDRESS OF MESSAGE TO BE TYPED
                                                            ;"LAST ADDRESS:"
                                ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $RDOCT ROUTINE
                                ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
005030  013746  177776          MOV     @#PSW,  -(SP)       ;PUT THE PROCESSOR STATUS ON THE STACK
005034  004767  016026          JSR     PC,     $RDOCT      ;GO TO THE SUBROUTINE
005040  005716                  TST     (SP)                ;CHECK IF ADR 0 SELECTED (DEFAULT).
005042  001010                  BNE     11$                 ;BR IF NOT 0 (DEFAULT)
005044  005767  016166          TST     $HIOCT              ;CHECK HI BITS.
005050  001025                  BNE     11$                 ;BR IF NOT 0 (DEFAULT).
005052  016716  174104          MOV     $TMP2,  (SP)        ;SET UP DEFAULT LAST ADR.
005056  016767  174102  016152  MOV     $TMP3,  $HIOCT
005064  012667  174502  11$:    MOV     (SP)+,  LSTADR      ;GET THE DATA.
005070  020467  016142          CMP     R4,     $HIOCT      ;CHECK FOR LAST ADR BELOW FIRST ADR.
005074  101352                  BHI     10$                 ;BB IF LAST BELOW FIRST.
005076  103403                  BLO     12$                 ;BR IF LAST ABOVE FIRST.
005100  021667  174466          CMP     (SP),   LSTADR      ;CHECK FOR LAST BELOW FIRST.
005104  101346                  BHI     10$                 ;BR IF LAST BELOW FIRST.
005106  032716  017777  12$:    BIT     #MASK4K,(SP)        ;CHECK IF FIRST ADR ON BANK BOUNDRY.
005112  001404                  BEQ     13$                 ;BR IF ON BOUNDRY.
```

```
 1902  005114  010067  174446                    MOV     R0,     FADMAP  ;SET UP FIRST ADDRESS MAP.
 1903  005120  010167  174444                    MOV     R1,     FADMAP+2
 1904  005124  050067  174402          13$:      BIS     R0,     SAVTST  ;SET FLAG IN TEST MAP...LO 64K.
 1905  005130  050167  174400                    BIS     R1,     SAVTST+2;...HI 64K.
 1906  005134  020367  016076          14$:      CMP     R3,     $HIOCT  ;CHECK FOR PAST LAST ADR.
 1907  005140  103404                            BLO     15$             ;BR IF BELOW LAST ADR.
 1908  005142  101020                            BHI     16$             ;BR IF GONE PAST LAST ADR.
 1909  005144  020267  174422                    CMP     R2,     LSTADR  ;CHECK FOR PAST LAST ADR.
 1910  005150  101015                            BHI     16$             ;BR IF GONE PAST LAST ADR.
 1911  005152  062702  020000          15$:      ADD     #20000, R2      ;UPDATE ADDRESS POINTER.
 1912  005156  005503                            ADC     R3              ;...HI BITS.
 1913  005160  006300                            ASL     R0              ;UPDATE BANK POINTER...LO 64K.
 1914  005162  006101                            ROL     R1              ;...HI 64K.
 1915  005164  100415                            BMI     17$             ;BR IF OVERFLOW.
 1916  005166  030067  174330                    BIT     R0,     MEMMAP  ;CHECK IF THIS BANK EXISTS.
 1917  005172  001354                            BNE     13$             ;BR IF BANK EXISTS.
 1918  005174  030167  174324                    BIT     R1,     MEMMAP+2;CHECK IF THIS BANK EXISTS.
 1919  005200  001351                            BNE     13$             ;BR IF BANK EXISTS.
 1920  005202  000754                            BR      14$             ;BR IF BANK DOESN'T EXIST.
 1921  005204  030067  174312          16$:      BIT     R0,     MEMMAP  ;CHECK IF THIS BANK EXISTS.
 1922  005210  001010                            BNE     20$             ;BR IF IT EXISTS.
 1923  005212  030167  174306                    BIT     R1,     MEMMAP+2;CHECK IF THIS BANK EXISTS.
 1924  005216  001005                            BNE     20$             ;BR IF IT EXISTS.
 1925  005220  005726                  17$:      TST     (SP)+           ;ADJUST THE STACK.
 1926  005222  004567  016012                    JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 1927  005226  026000                            .WORD   BADADR          ;ADDRESS OF MESSAGE TO BE TYPED
 1928                                                                    ;"?ADDRESS IN UNMAPPED BANK?"
 1929  005230  000606                            BR      MANUAL          ;LOOP BACK TO THE BEGINNING.
 1930  005232  010067  174342          20$:      MOV     R0,     LADMAP  ;SET UP MAP FOR LAST ADDRESS.
 1931  005236  010167  174340                    MOV     R1,     LADMAP+2
 1932  005242  005767  173340          21$:      TST     MMAVA           ;CHECK FOR MEMORY MANAGEMENT.
 1933  005246  001404                            BEQ     22$             ;BR IF NO MEM MGMT.
 1934  005250  042716  160000                    BIC     #160000,(SP)    ;ADJUST FSTADR TO VITRUAL BANK 0.
 1935  005254  062716  040000                    ADD     #40000, (SP)    ;...TO VIRTUAL BANK 2.
 1936  005260  012667  174274          22$:      MOV     (SP)+,  FSTADR  ;SAVE FISRT ADDRESS OFF THE STACK.
 1937  005264                          30$:
 1938  005264  004567  015750                    JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 1939  005270  026035                            .WORD   CONST           ;ADDRESS OF MESSAGE TO BE TYPED
 1940                                                                    ;"SELECT CONSTANT: "
 1941                                  ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $RDOCT ROUTINE
 1942                                  ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
 1943  005272  013746  177776                    MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
 1944  005276  004767  015564                    JSR     PC,     $RDOCT  ;GO TO THE SUBROUTINE
 1945  005302  012667  174300                    MOV     (SP)+,  .CONST  ;SAVE THE CONSTANT
 1946  005306  005767  173274          MANUL2:   TST     MMAVA           ;CHECK IF MEM MGMT IS AVAILABLE.
 1947  005312  001406                            BEQ     31$             ;BR IF NO MEM MGMT.
 1948  005314  042767  160000  174250            BIC     #160000,LSTADR  ;ADJUST LSTADR TO VIRTUAL BANK 0.
 1949  005322  062767  040000  174242            ADD     #40000, LSTADR  ;...VIRTUAL BANK 2.
 1950  005330  062767  000002  174234  31$:      ADD     #2,     LSTADR  ;ADJUST LAST ADDRESS UP ONE WORD.
 1951  005336  042767  000001  174226            BIC     #BIT0,  LSTADR  ;MAKE SURE IT IS A WORD ADDRESS.
 1952  005344  032767  017777  174220            BIT     #MASK4K,LSTADR  ;CHECK IF LAST ADR IS ON BANK BOUNDRY.
 1953  005352  001004                            BNE     START1          ;BR IF NOT ON BOUNDRY.
 1954  005354  005067  174220                    CLR     LADMAP          ;CLEAR OUT THE LAST ADDRESS MAP.
 1955  005360  005067  174216                    CLR     LADMAP+2
 1956
```

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

 1963  005364  016706  174220          START1: MOV     .STACK, SP      ;SET STACK POINTER
 1964  005370  012767  005364  173510          MOV     #START1,$LPADR  ;INIT LOOP ADDRESS.
 1965  005376  066767  173176  173502          ADD     RELOCF, $LPADR
 1966  005404  004767  012344                  JSR     PC,     MAMF    ;SET UP MEMORY PARITY ERROR VECTOR
 1967  005410  005767  173172                  TST     MMAVA           ;CHECK FOR MEMORY MANAGEMENT AVAILABLE.
 1968  005414  001406                          BEQ     TST1            ;BRANCH IF NO MEM MGMT.
 1969  005416  032737  000001  177572          BIT     #BIT0,  @#SR0   ;CHECK IF MEM MGMT ENABLED.
 1970  005424  001002                          BNE     TST1            ;BR IF MEM MGMT ENABLED.
 1971  005426  004767  007150                  JSR     PC,     MMINIT  ;SET UP MEM MGMT REGISTERS.
```

```
                                        .SBTTL SECTION 1:       MEMORY ADDRESS TESTS
                                        ;*********************************************************************
                                        ;*TEST 1        WRITE VALUE OF MEMORY ADDRESS INTO MEMORY
                                        ;*      R0 = DATA WRITTEN INTO MEMORY (SHOULD BE)
                                        ;*      R1 = DATA READ FROM MEMORY (WAS)
                                        ;*      R2 = VIRTUAL ADDRESS
                                        ;*      R3 = NOT USED
                                        ;*      R4 = NOT USED
                                        ;*      R5 = BLOCK BOUNDRY BIT MASK.
                                        ;*********************************************************************
005432                                  TST1:
005432  004567  013410                          JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
005436  000001                                  .WORD   1               ;MINIMUM BLOCK SIZE OF 1 WORDS
                                                                        ; REQUIRED FOR THIS TEST.
005440  000167  006702                          JMP     TST32           ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                                                                        ; AVAILABLE FOR TEST.
005444  012767  000001  173534                  MOV     #$TN-1,$TESTN   ;;SET TEST NUMBER IN  MAIL BOX
                                        ;*  UPWARDS WORD ADDRESSING.
005452  004467  007252                          JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
005456  004767  010674                  1$:     JSR     PC,     PHYADR  ;GET PHYSICAL ADDRESS INTO R0
005462  010012                          2$:     MOV     R0,     (R2)    ;WRITE VALUE OF ADDRESS INTO ADDRESS
005464  012201                                  MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
005466  020001                                  CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
005470  001405                                  BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
005472  004767  013106                  64$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
005476  004767  014334                          JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
005502  000002                                  .WORD   2               ;ERROR TYPE CODE.
005504                                  65$:
005504  062700  000002                          ADD     #2,     R0      ;ADD #2 TO PHYSICAL ADDRESS
005510  030502                                  BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
005512  001363                                  BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
005514  004767  007766                          JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.

                                        ;*  CHECK THAT VALUE OF MEMORY ADDRESS WAS WRITTEN CORRECTLY
                                        ;*  DOWNWARDS WORD ADDRESSING.
005520  004467  007642                          JSR     R4,     INITDN  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
005524  004767  010626                  3$:     JSR     PC,     PHYADR  ;GET PHYSICAL ADDRESS INTO R0
005530  162700  000002                  4$:     SUB     #2,     R0      ;DEC DATA BY 2
005534  014201                                  MOV     -(R2),  R1      ;GET THE DATA FROM MEMORY
005536  020001                                  CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
005540  001405                                  BEQ     67$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
005542  004767  013012                  66$:    JSR     PC,     SPRNT0  ;SET UP VALUES FOR ERROR PRINTING.
005546  004767  014264                          JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
005552  000002                                  .WORD   2               ;ERROR TYPE CODE.
005554                                  67$:
005554  030502                                  BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
005556  001364                                  BNE     4$              ;BRANCH IF MORE IN CURRENT BLOCK.
005560  004767  010412                          JSR     PC,     MMDOWN  ;FIND NEXT BLOCK AND LOOP TO $TAG1.
```

```
2020                                ;*****************************************************************
2021                                ;*TEST 2        WRITE VALUE OF MEMORY ADDRESS INTO MEMORY
2022                                ;*      R0 = DATA WRITTEN INTO MEMORY (SHOULD BE)
2023                                ;*      R1 = DATA READ FROM MEMORY (WAS)
2024                                ;*      R2 = VIRTUAL ADDRESS
2025                                ;*      R3 = NOT USED
2026                                ;*      R4 = NOT USED
2027                                ;*      R5 = BLOCK BOUNDRY BIT MASK.
2028                                ;*****************************************************************
2029  005564                        TST2:
2030  005564  004567  013256               JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
2031  005570  000000                       .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
2032  005572  012767  000002  173406       MOV     #$TN-1,$TESTN   ;;SET TEST NUMBER IN  MAIL BOX
2033                                ;*  UPWARDS BYTE ADDRESSING.
2034  005600  004467  007124               JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2035  005604  004767  010546        1$:    JSR     PC,     PHYADR  ;GET PHYSICAL ADDRESS INTO R0
2036  005610  110022                2$:    MOVB    R0,     (R2)+   ;WRITE VALUE OF ADDRESS INTO ADDRESS
2037  005612  005200                       INC     R0              ;ADD ONE TO PHYSICAL ADDRESS
2038  005614  030502                       BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
2039  005616  001374                       BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
2040  005620  004767  007662               JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
2041
2042                                ;*  CHECK THAT VALUE OF MEMORY ADDRESS WAS WRITTEN CORRECTLY
2043                                ;*  DOWNWARDS BYTE ADDRESSING.
2044  005624  004467  007536               JSR     R4,     INITDN  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2045  005630  004767  010522        3$:    JSR     PC,     PHYADR  ;GET PHYSICAL ADDRESS INTO R0
2046  005634  005300                4$:    DEC     R0              ;DEC DATA BY 1
2047  005636  114201                       MOVB    -(R2),  R1      ;GET THE DATA FROM MEMORY
2048  005640  120001                       CMPB    R0,     R1      ;CHECK THE DATA...LO BYTE ONLY VALID.
2049  005642  001405                       BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2050  005644  004767  012710        64$:   JSR     PC,     SPRNTO  ;SET UP VALUES FOR ERROR PRINTING.
2051  005650  004767  014162               JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2052  005654  000003                       .WORD   3               ;ERROR TYPE CODE.
2053  005656                        65$:
2054  005656  030502                       BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
2055  005660  001365                       BNE     4$              ;BRANCH IF MORE IN CURRENT BLOCK.
2056  005662  004767  010310               JSR     PC,     MMDOWN  ;FIND NEXT BLOCK AND LOOP TO $TAG1.
2057
2058                                ;*****************************************************************
2059                                ;*TEST 3        WRITE 1'S COMPLEMENT VALUE OF ADDRESS INTO ADDRESS.
2060                                ;*      R0 = DATA WRITTEN INTO MEMORY (SHOULD BE)
2061                                ;*      R1 = DATA READ FROM MEMORY (WAS)
2062                                ;*      R2 = VIRTUAL ADDRESS
2063                                ;*      R3 = NOT USED
2064                                ;*      R4 = NOT USED
2065                                ;*      R5 = BLOCK BOUNDRY BIT MASK.
2066                                ;*****************************************************************
2067  005666                        TST3:
2068  005666  004567  013154               JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
2069  005672  000000                       .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
2070  005674  012767  000003  173304       MOV     #$TN-1,$TESTN   ;;SET TEST NUMBER IN  MAIL BOX
2071                                ;*  DOWNWARDS WORD ADDRESSING.
2072  005702  004467  007460               JSR     R4,     INITDN  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2073  005706  004767  010444        1$:    JSR     PC,     PHYADR  ;GET PHYSICAL ADDRESS INTO R0
2074  005712  005100                       COM     R0              ;COMPLEMENT THE ADR
2075  005714  062700  000002        2$:    ADD     #2,     R0      ;+2 TO DATA--ADR GOES DOWN SO COM GOES UP
```

```
                                        MOV     R0,     -(R2)   ;PUT DATA INTO MEMORY
                                        BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
                                        BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
                                        JSR     PC,     MMDOWN  ;FIND NEXT BLOCK AND LOOP TO 1$.

                                ;*  CHECK COMPLEMENT DATA WRITTEN DOWN
                                ;*  UPWARDS WORD ADDRESSING.
                                        JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
                                3$:     JSR     PC,     PHYADR  ;GET PHYSICAL ADDRESS INTO R0
                                        COM     R0              ;COMPLEMENT IT
                                4$:
                                        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
                                        BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
                                        BNE     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 4K 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 BOUNDRY BIT MASK.
                                ;*****************************************************************
006000                          TST4:
006000  004567  013042                  JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
006004  000000                          .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
006006  012767  000004  173172          MOV     #$TN-1,$TESTN   ;;SET TEST NUMBER IN  MAIL BOX
                                ;*  UPWARDS BYTE ADDRESSING.
006014  004467  00671C                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
006020  004767  010406          1$:     JSR     PC,     BANKNO  ;GET THE BANK NUMBER INTO R0
006024  110022                  2$:     MOVB    R0,     (R2)+   ;WRITE BANK # INTO ALL ADDRESSES
006026  030502                          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
006030  001375                          BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
006032  004767  007450                  JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.

                                ;*  CHECK THAT DATA WRITTEN ABOVE CAN BE READ
                                ;*  UPWARDS BYTE ADDRESSING.
006036  004467  006666                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
006042  004767  010364          3$:     JSR     PC,     BANKNO  ;GET THE BANK NUMBER INTO R0
006046  112201                  4$:     MOVB    (R2)+,  R1      ;READ THE DATA OUT OF MEMORY
006050  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
006052  001405                          BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
006054  004767  012506          64$:    JSR     PC,     SPRNT1  ;SET UP VALUES FOR ERROR PRINTING.
006060  004767  013752                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
006064  000003                          .WORD   3               ;ERROR TYPE CODE.
006066                          65$:
006066  030502                          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
```

```
2132  006070  001366                  BNE     4$              ;BRANCH IF MORE IN CURRENT BLOCK.
2133  006072  004767  007410          JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 3$.
2134
2135                          ;*****************************************************************
2136                          ;*TEST 5        WRITE 1'S COMPLEMENT OF BANK #.
2137                          ;*      R0 = DATA WRITTEN INTO MEMORY (SHOULD BE)
2138                          ;*      R1 = DATA READ FROM MEMORY (WAS)
2139                          ;*      R2 = VIRTUAL ADDRESS
2140                          ;*      R3 = NOT USED
2141                          ;*      R4 = NOT USED
2142                          ;*      R5 = BLOCK BOUNDRY BIT MASK.
2143                          ;*****************************************************************
2144  006076                  TST5:
2145  006076  004567  012744          JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
2146  006102  000000                  .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
2147  006104  012767  000005  173074  MOV     #$TN-1,$TESTN   ;;SET TEST NUMBER IN  MAIL BOX
2148                          ;*  DOWNWARDS BYTE ADDRESSING.
2149  006112  004467  007250          JSR     R4,     INITDN  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2150  006116  004767  010310  1$:     JSR     PC,     BANKNO  ;GET THE BANK NUMBER INTO R0
2151  006122  005100                  COM     R0              ;1'S COMPLEMENT OF BANK #
2152  006124  110042          2$:     MOVB    R0,     -(R2)   ;PUT 1'S COM OF BANK # INTO MEMORY
2153  006126  030502                  BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
2154  006130  001375                  BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
2155  006132  004767  010040          JSR     PC,     MMDOWN  ;FIND NEXT BLOCK AND LOOP TO 1$.
2156
2157                          ;*  CHECK THAT DATA WRITTEN CAN BE READ.
2158                          ;*  DOWNWARDS BYTE ADDRESSING.
2159  006136  004467  007224          JSR     R4,     INITDN  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2160  006142  004767  010264  3$:     JSR     PC,     BANKNO  ;GET THE BANK # INTO R0
2161  006146  005100                  COM     R0              ;SET 1'S COMPLEMENT OF BANK #
2162  006150  114201          4$:     MOVB    -(R2),  R1      ;READ DATA OUT OF MEMORY
2163  006152  020001                  CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2164  006154  001405                  BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2165  006156  004767  012376  64$:     JSR     PC,     SPRNTO  ;SET UP VALUES FOR ERROR PRINTING.
2166  006162  004767  013650          JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2167  006166  000003                  .WORD   3               ;ERROR TYPE CODE.
2168  006170                  65$:
2169  006170  030502                  BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
2170  006172  001366                  BNE     4$              ;BRANCH IF MORE IN CURRENT BLOCK.
2171  006174  004767  007776          JSR     PC,     MMDOWN  ;FIND NEXT BLOCK AND LOOP TO $TAG1.
```

```
                                .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 BOUNDRY BIT MASK.
                                ;*****************************************************************
006200                          TST6:
006200  004567  012642                  JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
006204  000000                          .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
006206  012767  000006  172772          MOV     #$TN-1,$TESTN   ;;SET TEST NUMBER IN  MAIL BOX
006214  016700  173366          TST6A:  MOV     .CONST, R0      ;GET USER CONSTANT
006220  004467  006504                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
006224  010022                  1$:     MOV     R0,     (R2)+   ;WRITE CONSTANT INTO MEMORY.
006226  030502                          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
006230  001375                          BNE     1$              ;BRANCH IF MORE IN CURRENT BLOCK.
006232  004767  007250                  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.
                                ;*****************************************************************
006236                          TST7:
006236  004567  012604                  JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
006242  000000                          .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
006244  012767  000007  172734          MOV     #$TN-1,$TESTN   ;;SET TEST NUMBER IN  MAIL BOX
006252  016700  173330                  MOV     .CONST, R0      ;GET USER CONSTANT
006256  004467  006446                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
006262                          1$:
006262  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
006264  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
006266  001405                          BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
006270  004767  012310          64$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
006274  004767  013536                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
006300  000004                          .WORD   4               ;ERROR TYPE CODE.
006302                          65$:
006302  030502                          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
006304  001366                          BNE     1$              ;BRANCH IF MORE IN CURRENT BLOCK.
006306  004767  007174                  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.
006312  032777  000400  172616          BIT     #SW08,  @SWR    ;CHECK THAT LOOP ON TEST BIT SET
006320  001416                          BEQ     TST10           ;BRANCH IF NOT LOOP ON TEST
006322  017746  172610                  MOV     @SWR,   -(SP)   ;GET SWITCH REGISTER DATA.
006326  042716  177740                  BIC     #177740,(SP)    ;CLEAR NON-TEST-NUMBER SWITCHS.
006332  022726  000006                  CMP     #6,     (SP)+   ;CHECK IF TEST 6 IN SWITCHES.
006336  001007                          BNE     TST10           ;BRANCH IF NOT TEST 6
```

```
2228  006340  162767  000001  172534          SUB     #1,     $TSTNM  ;RESET TEST NUM
2229  006346  162767  000036  172532          SUB     #TST7-TST6,$LPADR ;RESET LOOP ADR
2230  006354  000717                          BR      TST6A           ;GO TO TEST 6
2231
2232                                  ;*****************************************************************
2233                                  ;*TEST 10       WORSE CASE NOISE (PARITY) WORD TESTING
2234                                          ;*  CHECK MEMORY WITH A SERIES OF PATTERNS
2235                                  ;*****************************************************************
2236  006356                          TST10:
2237  006356  004567  012464                 JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
2238  006362  000000                         .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
2239  006364  012767  000010  172614         MOV     #$TN-1,$TESTN   ;;SET TEST NUMBER IN  MAIL BOX
2240  006372  016704  173232                 MOV     .MPPAT, R4      ;INITIALIZE PATTERN TABLE POINTER
2241  006376  004767  011452          1$:    JSR     PC,     CKPMER  ;CHECK FOR NON-TRAP PARITY MEMORY ERRORS.
2242  006402  012400                         MOV     (R4)+,  R0      ;GET THE DATA PATTERN.
2243  006404  001420                         BEQ     TST11           ;;BR IF END OF TABLE.
2244  006406  004467  006316                 JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2245  006412  010012                  2$:    MOV     R0,     (R2)    ;PUT DATA PATTERN INTO MEMORY.
2246  006414  012201                         MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2247  006416  020001                         CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2248  006420  001405                         BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2249  006422  004767  012156          64$:   JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2250  006426  004767  013404                 JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2251  006432  000004                         .WORD   4               ;ERROR TYPE CODE.
2252  005434                          65$:
2253  006434  030502                         BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
2254  006436  001365                         BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
2255  006440  004767  007042                 JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 2$.
2256  006444  000754                         BR      1$              ;BR BACK TO DO NEXT PATTERN
```

```
2257                                    ;*****************************************************************
2258                                    ;*TEST 11       ROTATE A "0" BIT THROUGH A FIELD OF ONES.
2259                                    ;*****************************************************************
2260  006446                            TST11:
2261  006446  004567  012374                   JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
2262  006452  000000                            .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
2263  006454  012767  000011  172524           MOV     #$TN-1,$TESTN   ;;SET TEST NUMBER IN  MAIL BOX
2264  006462  012700  177777                   MOV     #-1,    R0      ;SET CHECK WORD
2265  006466  004767  010000                   JSR     PC,     SETCON  ;PUT THE CONTENTS OF R0 IN ALL MEMORY.
2266  006472  004467  006232                   JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2267  006476  000241                    1$:     CLC                     ;CLEAR CARRY BIT IN PSW
2268  006500  004767  010006                   JSR     PC,     ROTATE
2269  006504  016201  177776                   MOV     -2(R2), R1      ;GET RESULT
2270  006510  103402                            BCS     63$             ;BRANCH IF 'C' BIT WAS SET
2271  006512  020001                            CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2272  006514  001405                            BEQ     64$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2273  006516  004767  012062            63$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2274  006522  004767  013310                   JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2275  006526  000005                            .WORD   5               ;ERROR TYPE CODE.
2276  006530                            64$:
2277  006530  030502                            BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
2278  006532  001361                            BNE     1$              ;BRANCH IF MORE IN CURRENT BLOCK.
2279  006534  004767  006746                   JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
2280
2281                                    ;*****************************************************************
2282                                    ;*TEST 12       ROTATE A "1" BIT THROUGH A FIELD OF ZEROS
2283                                    ;*****************************************************************
2284  006540                            TST12:
2285  006540  004567  012302                   JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
2286  006544  000000                            .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
2287  006546  012767  000012  172432           MOV     #$TN-1,$TESTN   ;;SET TEST NUMBER IN  MAIL BOX
2288  006554  005000                            CLR     R0              ;SET CHECK WORD
2289  006556  004767  007710                   JSR     PC,     SETCON  ;PUT THE CONTENTS OF R0 IN ALL MEMORY
2290  006562  004467  006142                   JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2291  006566  000261                    1$:     SEC                     ;SET 'C' BIT IN PSW
2292  006570  004767  007716                   JSR     PC,     ROTATE  ;GO ROTATE '1' BIT
2293  006574  016201  177776                   MOV     -2(R2), R1      ;GET RESULT
2294  006600  103002                            BCC     63$             ;BRANCH IF 'C' IS CLEAR
2295  006602  020001                            CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2296  006604  001405                            BEQ     64$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2297  006606  004767  011772            63$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2298  006612  004767  013220                   JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2299  006616  000005                            .WORD   5               ;ERROR TYPE CODE.
2300  006620                            64$:
2301  006620  030502                            BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
2302  006622  001361                            BNE     1$              ;BRANCH IF MORE IN CURRENT BLOCK.
2303  006624  004767  006656                   JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
```

```
2304                                    ;*********************************************************************
2305                                    ;*TEST 13        1 XOR 8 TEST PATTERN
2306                                    ;*********************************************************************
2307  006630                            TST13:
2308  006630  004567  012212                   JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
2309  006634  000377                           .WORD   377             ;MINIMUM BLOCK SIZE OF 128. WORDS
2310                                                                    ; REQUIRED FOR THIS TEST.
2311  006636  000167  000442                   JMP     TST14           ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
2312                                                                    ; AVAILABLE FOR TEST.
2313  006642  012767  000013  172336           MOV     #$TN-1,$TESTN   ;;SET TEST NUMBER IN  MAIL BOX
2314  006650  012700  177777                   MOV     #-1,    R0      ;SET UP CHECK WORD.
2315  006654  005003                           CLR     R3              ;SET UP COM DATA REG
2316  006656  004467  006046                   JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2317  006662  005100                    1$:    COM     R0
2318  006664  005103                           COM     R3
2319  006666  012704  000010                   MOV     #8.,R4          ;SET 128. WORD COUNTER
2320  006672  010022                    2$:    MOV     R0,(R2)+        ;WRITE 128. WORDS
2321  006674  010322                           MOV     R3,(R2)+
2322  006676  010022                           MOV     R0,(R2)+
2323  006700  010322                           MOV     R3,(R2)+
2324
2325  006702  010022                           MOV     R0,(R2)+
2326  006704  010322                           MOV     R3,(R2)+
2327  006706  010022                           MOV     R0,(R2)+
2328  006710  010322                           MOV     R3,(R2)+
2329
2330  006712  010022                           MOV     R0,(R2)+
2331  006714  010322                           MOV     R3,(R2)+
2332  006716  010022                           MOV     R0,(R2)+
2333  006720  010322                           MOV     R3,(R2)+
2334
2335  006722  010022                           MOV     R0,(R2)+
2336  006724  010322                           MOV     R3,(R2)+
2337  006726  010022                           MOV     R0,(R2)+
2338  006730  010322                           MOV     R3,(R2)+
2339
2340  006732  005304                           DEC     R4              ;DECREMENT 128. WORD COUNTER
2341  006734  001356                           BNE     2$
2342  006736  030502                           BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
2343  006740  001350                           BNE     1$              ;BRANCH IF MORE IN CURRENT BLOCK.
2344  006742  004767  006540                   JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
2345
2346                                  ;*********************************************************************
2347                                  ;*  CHECK 1 XOR 8 TEST PATTERN WRITTEN ABOVE.
2348                                  ;*********************************************************************
2349  006746  005000                           CLR     R0              ;CLEAR TEST WORD
2350  006750  004467  005754                   JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2351  006754  012704  000040            11$:   MOV     #32.,   R4      ;SET 128. WORD COUNTER
2352  006760                             12$:
2353  006760  012201                           MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2354  006762  020001                           CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2355  006764  001405                           BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2356  006766  004767  011612            64$:   JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2357  006772  004767  013040                   JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2358  006776  000006                           .WORD   6               ;ERROR TYPE CODE.
2359  007000                             65$:
```

```
2360  007000  005100                        COM     R0
2361  007002  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2362  007004  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2363  007006  001405                        BEQ     67$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2364  007010  004767  011570        66$:    JSR     PC,     $PRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2365  007014  004767  013016                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2366  007020  000006                        .WORD   6               ;ERROR TYPE CODE.
2367  007022                        67$:
2368  007022  005100                        COM     R0
2369  007024  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2370  007026  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2371  007030  001405                        BEQ     69$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2372  007032  004767  011546        68$:    JSR     PC,     $PRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2373  007036  004767  012774                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2374  007042  000006                        .WORD   6               ;ERROR TYPE CODE.
2375  007044                        69$:
2376  007044  005100                        COM     R0
2377  007046  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2378  007050  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2379  007052  001405                        BEQ     71$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2380  007054  004767  011524        70$:    JSR     PC,     $PRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2381  007060  004767  012752                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2382  007064  000006                        .WORD   6               ;ERROR TYPE CODE.
2383  007066                        71$:
2384  007066  005100                        COM     R0              ;COMPLEMENT TEST DATA
2385  007070  005304                        DEC     R4              ;DECREMENT 128. WORD COUNTER
2386  007072  001332                        BNE     12$
2387  007074  005100                        COM     R0              ;COMPLEMENT TEST DATA
2388  007076  030502                        BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
2389  007100  001325                        BNE     11$             ;BRANCH IF MORE IN CURRENT BLOCK.
2390  007102  004767  006400                JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 11$.
2391
2392                        ;*****************************************************************
2393                        ;*  COMPLEMENT 1 XOR 8 PATTERN WRITTEN ABOVE.
2394                        ;*****************************************************************
2395  007106  004467  005616                JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2396  007112  012704  000040        21$:    MOV     #32.,   R4      ;SET UP 128. WORD BLOCK COUNTER
2397  007116  005122                22$:    COM     (R2)+           ;COMPLEMENT PATTERN
2398  007120  005122                        COM     (R2)+
2399  007122  005122                        COM     (R2)+
2400  007124  005122                        COM     (R2)+
2401  007126  005304                        DEC     R4              ;CHECK FOR 128. WORDS DONE.
2402  007130  001372                        BNE     22$             ;BRANCH IF MORE
2403  007132  030502                        BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
2404  007134  001366                        BNE     21$             ;BRANCH IF MORE IN CURRENT BLOCK.
2405  007136  004767  006344                JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 21$.
2406
2407                        ;*****************************************************************
2408                        ;*  CHECK COMPLEMENTED 1 XOR 8 TEST PATTERN.
2409                        ;*****************************************************************
2410  007142  012700  177777                MOV     #-1,    R0      ;INITIALIZE TEST WORD.
2411  007146  004467  005556                JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2412  007152  012704  000040        31$:    MOV     #32.,   R4      ;SET 128. WORD COUNTER
2413  007156                        32$:
2414  007156  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2415  007160  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
```

```
2416  007162  001405                        BEQ     73$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2417  007164  004767  011414        72$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2418  007170  004767  012642                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2419  007174  000006                        .WORD   6               ;ERROR TYPE CODE.
2420  007176                        73$:
2421  007176  005100                        COM     R0
2422  007200  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2423  007202  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2424  007204  001405                        BEQ     75$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2425  007206  004767  011372        74$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2426  007212  004767  012620                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2427  007216  000006                        .WORD   6               ;ERROR TYPE CODE.
2428  007220                        75$:
2429  007220  005100                        COM     R0
2430  007222  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2431  007224  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2432  007226  001405                        BEQ     77$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2433  007230  004767  011350        76$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2434  007234  004767  012576                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2435  007240  000006                        .WORD   6               ;ERROR TYPE CODE.
2436  007242                        77$:
2437  007242  005100                        COM     R0
2438  007244  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2439  007246  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2440  007250  001405                        BEQ     79$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2441  007252  004767  011326        78$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2442  007256  004767  012554                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2443  007262  000006                        .WORD   6               ;ERROR TYPE CODE.
2444  007264                        79$:
2445  007264  005100                        COM     R0              ;COMPLEMENT TEST DATA
2446  007266  005304                        DEC     R4              ;DECREMENT 128. WORD COUNTER
2447  007270  001332                        BNE     32$
2448  007272  005100                        COM     R0              ;COMPLEMENT TEST DATA
2449  007274  030502                        BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
2450  007276  001325                        BNE     31$             ;BRANCH IF MORE IN CURRENT BLOCK.
2451  007300  004767  006202                JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 31$.
```

```
2452                                    ;*****************************************************************
2453                                    ;*TEST 14        3 XOR 9 TEST PATTERN.
2454                                    ;*****************************************************************
2455  007304                            TST14:
2456  007304  004567  011536                   JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
2457  007310  000777                           .WORD   777             ;MINIMUM BLOCK SIZE OF 256. WORDS
2458                                                                    ; REQUIRED FOR THIS TEST.
2459  007312  000167  000320                   JMP     TST15           ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
2460                                                                    ; AVAILABLE FOR TEST.
2461  007316  012767  000014  171662           MOV     #$TN-1,$TESTN   ;;SET TEST NUMBER IN  MAIL BOX
2462  007324  005000                    .3X9:  CLR     R0              ;SET UP TEST DATA
2463  007326  012703  177777                   MOV     #-1,    R3      ;SET COM DATA REG
2464  007332  004467  005372                   JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2465  007336  004767  007216            1$:    JSR     PC,     W3X9    ;WRITE 256. WORD BLOCK WITH 3 XOR 9 PAT.
2466  007342  030502                           BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
2467  007344  001374                           BNE     1$              ;BRANCH IF MORE IN CURRENT BLOCK.
2468  007346  004767  006134                   JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
2469
2470                                    ;*****************************************************************
2471                                    ;*  CHECK 3 XOR 9 TEST PATTERN WRITTEN ABOVE
2472                                    ;*****************************************************************
2473  007352  005000                           CLR     R0              ;SET CHECK WORD
2474  007354  004467  005350                   JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2475  007360  012704  000100            11$:   MOV     #64.,   R4      ;SET 256. WORD COUNTER
2476  007364                            12$:
2477  007364  012201                           MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2478  007366  020001                           CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2479  007370  001405                           BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2480  007372  004767  011206            64$:   JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2481  007376  004767  012434                   JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2482  007402  000007                           .WORD   7               ;ERROR TYPE CODE.
2483  007404                            65$:
2484  007404  012201                           MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2485  007406  020001                           CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2486  007410  001405                           BEQ     67$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2487  007412  004767  011166            66$:   JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2488  007416  004767  012414                   JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2489  007422  000007                           .WORD   7               ;ERROR TYPE CODE.
2490  007424                            67$:
2491  007424  012201                           MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2492  007426  020001                           CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2493  007430  001405                           BEQ     69$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2494  007432  004767  011146            68$:   JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2495  007436  004767  012374                   JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2496  007442  000007                           .WORD   7               ;ERROR TYPE CODE.
2497  007444                            69$:
2498  007444  012201                           MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2499  007446  020001                           CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2500  007450  001405                           BEQ     71$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2501  007452  004767  011126            70$:   JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2502  007456  004767  012354                   JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2503  007462  000007                           .WORD   7               ;ERROR TYPE CODE.
2504  007464                            71$:
2505  007464  005100                           COM     R0              ;COMPLEMENT CHECK WORD
2506  007466  005304                           DEC     R4              ;DECREMENT 256. WORD COUNTER
2507  007470  001335                           BNE     12$
```

```
2508  007472  005100                          COM     R0              ;COMPLEMENT CHECK WORD
2509  007474  030502                          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
2510  007476  001330                          BNE     11$             ;BRANCH IF MORE IN CURRENT BLOCK.
2511  007500  004767  006002                  JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 11$.
2512
2513                                  ;*****************************************************************
2514                                  ;*  CHECK, COM, CHECK, COM, CHECK 3 XOR 9 PATTERN WRITTEN ABOVE.
2515                                  ;*****************************************************************
2516  007504  005000                          CLR     R0
2517  007506  004467  005216                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2518  007512  012704  000100          21$:    MOV     #64.,   R4      ;SET 256. WORD COUNTER
2519  007516  012703  000004          22$:    MOV     #4,     R3      ;SET 4 WORD COUNTER
2520  007522                          23$:
2521  007522  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2522  007524  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2523  007526  001405                          BEQ     73$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2524  007530  004767  011050          72$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2525  007534  004767  012276                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2526  007540  000007                          .WORD   7               ;ERROR TYPE CODE.
2527  007542                          73$:
2528  007542  005100                          COM     R0              ;COMPLEMENT CHECK WORD
2529  007544  005142                          COM     -(R2)           ;COMPLEMENT TEST DATA
2530  007546  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2531  007550  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2532  007552  001405                          BEQ     75$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2533  007554  004767  011024          74$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2534  007560  004767  012252                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2535  007564  000007                          .WORD   7               ;ERROR TYPE CODE.
2536  007566                          75$:
2537  007566  005100                          COM     R0              ;COMPLEMENT CHECK WORD
2538  007570  005142                          COM     -(R2)           ;COMPLEMENT TEST DATA
2539  007572  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2540  007574  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2541  007576  001405                          BEQ     77$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2542  007600  004767  011000          76$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2543  007604  004767  012226                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2544  007610  000007                          .WORD   7               ;ERROR TYPE CODE.
2545  007612                          77$:
2546  007612  005303                          DEC     R3              ;DECREMENT 4 WORD COUNTER
2547  007614  001342                          BNE     23$             ;BR IF NOT DONE.
2548  007616  005100                          COM     R0              ;COMPLEMENT CHECK WORD
2549  007620  005304                          DEC     R4              ;DECREMENT 256. WORD COUNTER
2550  007622  001335                          BNE     22$             ;BR IF NOT DONE.
2551  007624  005100                          COM     R0              ;COMPLEMENT CHECK WORD
2552  007626  030502                          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
2553  007630  001330                          BNE     21$             ;BRANCH IF MORE IN CURRENT BLOCK.
2554  007632  004767  005650                  JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 21$.
```

```
                                ;*****************************************************************
                                ;*TEST 15       COMPLEMENT 3 XOR 9 TEST PATTERN
                                ;*****************************************************************
007636                          TST15:
007636  004567  011204                  JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
007642  000777                          .WORD   777             ;MINIMUM BLOCK SIZE OF 256. WORDS
                                                                ; REQUIRED FOR THIS TEST.
007644  000167  000324                  JMP     TST16           ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                                                                ; AVAILABLE FOR TEST.
007650  012767  000015  171330          MOV     #STN-1,$TESTN   ;;SET TEST NUMBER IN  MAIL BOX
007656  012700  177777                  MOV     #-1,    R0      ;SET UP TEST DATA
007662  005003                          CLR     R3              ;SET COM DATA REG
007664  004467  005040                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
007670  004767  006664          1$:     JSR     PC,     W3X9    ;WRITE 256. WORD BLOCK WITH 3 XOR 9 PAT.
007674  030502                          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
007676  001374                          BNE     1$              ;BRANCH IF MORE IN CURRENT BLOCK.
007700  004767  005602                  JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.


                                ;*****************************************************************
                                ;*  CHECK COMPLEMENTED 3 XOR 9 TEST PATTERN WRITTEN ABOVE.
                                ;*****************************************************************
007704  012700  177777                  MOV     #-1,    R0      ;SET CHECK WORD
007710  004467  005014                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
007714  012704  000100          11$:    MOV     #64.,   R4      ;SET 256. WORD COUNTER
007720                          12$:
007720  012201                  MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
007722  020001                  CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
007724  001405                  BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
007726  004767  010652          64$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
007732  004767  012100                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
007736  000007                          .WORD   7               ;ERROR TYPE CODE.
007740                          65$:
007740  012201                  MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
007742  020001                  CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
007744  001405                  BEQ     67$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
007746  004767  010632          66$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
007752  004767  012060                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
007756  000007                          .WORD   7               ;ERROR TYPE CODE.
007760                          67$:
007760  012201                  MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
007762  020001                  CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
007764  001405                  BEQ     69$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
007766  004767  010612          68$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
007772  004767  012040                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
007776  000007                          .WORD   7               ;ERROR TYPE CODE.
010000                          69$:
010000  012201                  MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
010002  020001                  CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
010004  001405                  BEQ     71$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
010006  004767  010572          70$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
010012  004767  012020                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
010016  000007                          .WORD   7               ;ERROR TYPE CODE.
010020                          71$:
010020  005100                  COM     R0              ;COMPLEMENT CHECK WORD
010022  005304                  DEC     R4              ;DECREMENT 256. WORD COUNTER
```

```
2611  010024  001335                        BNE     12$
2612  010026  005100                        COM     R0                      ;COMPLEMENT CHECK WORD
2613  010030  030502                        BIT     R5,     R2              ;CHECK FOR END OF A BLOCK.
2614  010032  001330                        BNE     11$                     ;BRANCH IF MORE IN CURRENT BLOCK.
2615  010034  004767  005446                JSR     PC,     MMUP            ;FIND NEXT BLOCK AND LOOP TO 11$.
2616
2617                                ;*****************************************************************
2618                                ;*  CHECK, COM, CHECK, COM, CHECK COMPLEMENTED 3 XOR 9 PATTERN.
2619                                ;*****************************************************************
2620  010040  012700  177777                MOV     #-1,    R0              ;SET UP CHECK WORD.
2621  010044  004467  004660                JSR     R4,     INITMM          ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2622  010050  012704  000100        21$:    MOV     #64.,   R4              ;SET 256. WORD COUNTER
2623  010054  012703  000004        22$:    MOV     #4,     R3              ;SET 4 WORD COUNTER
2624  010060                         23$:
2625  010060  012201                        MOV     (R2)+,  R1              ;GET THE DATA FROM MEMORY UNDER TEST.
2626  010062  020001                        CMP     R0,     R1              ;COMPARE THE CHECK WORD WITH THE DATA READ.
2627  010064  001405                        BEQ     73$                     ;BRANCH OVER ERROR CALL IF GOOD DATA.
2628  010066  004767  010512        72$:    JSR     PC,     SPRNT2          ;SET UP VALUES FOR ERROR PRINTING.
2629  010072  004767  011740                JSR     PC,     $ERROR          ;*** ERROR *** (GO TYPE A MESSAGE)
2630  010076  000007                        .WORD   7                       ;ERROR TYPE CODE.
2631  010100                         73$:
2632  010100  005100                        COM     R0                      ;COMPLEMENT CHECK WORD
2633  010102  005142                        COM     -(R2)                   ;COMPLEMENT TEST DATA
2634  010104  012201                        MOV     (R2)+,  R1              ;GET THE DATA FROM MEMORY UNDER TEST.
2635  010106  020001                        CMP     R0,     R1              ;COMPARE THE CHECK WORD WITH THE DATA READ.
2636  010110  001405                        BEQ     75$                     ;BRANCH OVER ERROR CALL IF GOOD DATA.
2637  010112  004767  010466        74$:    JSR     PC,     SPRNT2          ;SET UP VALUES FOR ERROR PRINTING.
2638  010116  004767  011714                JSR     PC,     $ERROR          ;*** ERROR *** (GO TYPE A MESSAGE)
2639  010122  000007                        .WORD   7                       ;ERROR TYPE CODE.
2640  010124                         75$:
2641  010124  005100                        COM     R0                      ;COMPLEMENT CHECK WORD
2642  010126  005142                        COM     -(R2)                   ;COMPLEMENT TEST DATA
2643  010130  012201                        MOV     (R2)+,  R1              ;GET THE DATA FROM MEMORY UNDER TEST.
2644  010132  020001                        CMP     R0,     R1              ;COMPARE THE CHECK WORD WITH THE DATA READ.
2645  010134  001405                        BEQ     77$                     ;BRANCH OVER ERROR CALL IF GOOD DATA.
2646  010136  004767  010442        76$:    JSR     PC,     SPRNT2          ;SET UP VALUES FOR ERROR PRINTING.
2647  010142  004767  011670                JSR     PC,     $ERROR          ;*** ERROR *** (GO TYPE A MESSAGE)
2648  010146  000007                        .WORD   7                       ;ERROR TYPE CODE.
2649  010150                         77$:
2650  010150  005303                        DEC     R3                      ;DECREMENT 4 WORD COUNTER
2651  010152  001342                        BNE     23$                     ;BR IF NOT DONE.
2652  010154  005100                        COM     R0                      ;COMPLEMENT CHECK WORD
2653  010156  005304                        DEC     R4                      ;DECREMENT 256. WORD COUNTER
2654  010160  001335                        BNE     22$                     ;BR IF NOT DONE.
2655  010162  005100                        COM     R0                      ;COMPLEMENT CHECK WORD
2656  010164  030502                        BIT     R5,     R2              ;CHECK FOR END OF A BLOCK.
2657  010166  001330                        BNE     21$                     ;BRANCH IF MORE IN CURRENT BLOCK.
2658  010170  004767  005312                JSR     PC,     MMUP            ;FIND NEXT BLOCK AND LOOP TO 21$.
```

```
2659                                    ;*****************************************************************
2660                                    ;*TEST 16       MODIFIED 3 XOR 9 PATTERN FOR PARITY MEMORY
2661                                    ;*****************************************************************
2662  010174                            TST16:
2663  010174  004567  010646                   JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
2664  010200  000777                           .WORD   777             ;MINIMUM BLOCK SIZE OF 256. WORDS
2665                                                                    ; REQUIRED FOR THIS TEST.
2666  010202  000167  000616                   JMP     TST17           ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
2667                                                                    ; AVAILABLE FOR TEST.
2668  010206  012767  000016  170772           MOV     #$TN-1,$TESTN   ;;SET TEST NUMBER IN  MAIL BOX
2669  010214  012700  000401                   MOV     #401,   R0      ;SET UP PARITY "ALL ZEROS" PATTERN
2670  010220  012703  177777                   MOV     #-1,    R3      ;SET COM DATA REG
2671  010224  004467  004500                   JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2672  010230  004767  006324           1$:     JSR     PC,     W3X9    ;WRITE 256. WORD BLOCK WITH 3 XOR 9 PAT.
2673  010234  030502                           BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
2674  010236  001374                           BNE     1$              ;BRANCH IF MORE IN CURRENT BLOCK.
2675  010240  004767  005242                   JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
2676
2677                                    ;*****************************************************************
2678                                    ;*  CHECK PARITY 3 XOR 9 PATTERN WRITTEN ABOVE.
2679                                    ;*****************************************************************
2680  010244  012700  000401                   MOV     #401,   R0      ;RESET PARITY "ALL ZEROS" PATTERN.
2681  010250  012703  177777                   MOV     #-1,    R3      ;RESET PARITY ALL ONES PATTERN.
2682  010254  004467  004450                   JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2683  010260  012704  000100           11$:    MOV     #64.,   R4      ;SET 256. WORD COUNTER
2684  010264                           12$:
2685  010264  012201                           MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2686  010266  020001                           CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2687  010270  001405                           BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2688  010272  004767  010306           64$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2689  010276  004767  011534                   JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2690  010302  000007                           .WORD   7               ;ERROR TYPE CODE.
2691  010304                           65$:
2692  010304  012201                           MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2693  010306  020001                           CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2694  010310  001405                           BEQ     67$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2695  010312  004767  010266           66$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2696  010316  004767  011514                   JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2697  010322  000007                           .WORD   7               ;ERROR TYPE CODE.
2698  010324                           67$:
2699  010324  012201                           MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2700  010326  020001                           CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2701  010330  001405                           BEQ     69$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2702  010332  004767  010246           68$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2703  010336  004767  011474                   JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2704  010342  000007                           .WORD   7               ;ERROR TYPE CODE.
2705  010344                           69$:
2706  010344  012201                           MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2707  010346  020001                           CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2708  010350  001405                           BEQ     71$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2709  010352  004767  010226           70$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2710  010356  004767  011454                   JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2711  010362  000007                           .WORD   7               ;ERROR TYPE CODE.
2712  010364                           71$:
2713  010364  010046                           MOV     R0,     -(SP)   ;SAVE R0
2714  010366  010300                           MOV     R3,     R0      ;PUT R3 INTO R0
```

```
2715  010370  012603                        MOV     (SP)+,  R3      ;PUT SAVED R0 INTO R3
2716  010372  005304                        DEC     R4              ;COUNT 256. WORDS
2717  010374  001333                        BNE     12$             ;BRANCH IF MORE
2718  010376  010046                        MOV     R0,     -(SP)   ;SAVE R0
2719  010400  010300                        MOV     R3,     R0      ;PUT R3 INTO R0
2720  010402  012603                        MOV     (SP)+,  R3      ;PUT SAVED R0 INTO R3
2721  010404  030502                        BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
2722  010406  001324                        BNE     11$             ;BRANCH IF MORE IN CURRENT BLOCK.
2723  010410  004767  005072                JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 11$.
2724
2725                                ;*****************************************************************
2726                                ;*  CHECK, COM, CHECK, COM, CHECK PARITY 3 XOR 9 PATTERN.
2727                                ;*****************************************************************
2728  010414  012700  000401                MOV     #401,   R0      ;SET UP PARITY "ALL ZEROS" PATTERN.
2729  010420  012703  177777                MOV     #-1,    R3      ;SET UP ALL ONES PATTERN.
2730  010424  004467  004300                JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2731  010430  012704  000100        21$:    MOV     #64.,R4         ;SET 256. WORD COUNTER
2732  010434                        22$:
2733  010434  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2734  010436  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2735  010440  001405                        BEQ     73$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2736  010442  004767  010136        72$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2737  010446  004767  011364                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2738  010452  000007                        .WORD   7               ;ERROR TYPE CODE.
2739  010454                        73$:
2740  010454  005100                        COM     R0              ;COMPLEMENT CHECK WORD
2741  010456  005142                        COM     -(R2)           ;COMPLEMENT TEST DATA
2742  010460  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2743  010462  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2744  010464  001405                        BEQ     75$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2745  010466  004767  010112        74$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2746  010472  004767  011340                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2747  010476  000007                        .WORD   7               ;ERROR TYPE CODE.
2748  010500                        75$:
2749  010500  005100                        COM     R0              ;COMPLEMENT CHECK WORD
2750  010502  005142                        COM     -(R2)           ;RESTORE DATA
2751  010504  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2752  010506  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2753  010510  001405                        BEQ     77$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2754  010512  004767  010066        76$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2755  010516  004767  011314                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2756  010522  000007                        .WORD   7               ;ERROR TYPE CODE.
2757  010524                        77$:
2758  010524  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2759  010526  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2760  010530  001405                        BEQ     79$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2761  010532  004767  010046        78$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2762  010536  004767  011274                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2763  010542  000007                        .WORD   7               ;ERROR TYPE CODE.
2764  010544                        79$:
2765  010544  005100                        COM     R0              ;COMPLEMENT CHECK WORD
2766  010546  005142                        COM     -(R2)           ;COMPLEMENT TEST DATA
2767  010550  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2768  010552  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2769  010554  001405                        BEQ     81$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2770  010556  004767  010022        80$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
```

```
2771  010562  004767  011250            JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2772  010566  000007                    .WORD   7               ;ERROR TYPE CODE.
2773  010570                    81$:
2774  010570  005100                    COM     R0              ;COMPLEMENT CHECK WORD
2775  010572  005142                    COM     -(R2)           ;RESTORE DATA
2776  010574  012201                    MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2777  010576  020001                    CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2778  010600  001405                    BEQ     83$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2779  010602  004767  007776    82$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2780  010606  004767  011224            JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2781  010612  000007                    .WORD   7               ;ERROR TYPE CODE.
2782  010614                    83$:
2783  010614  012201                    MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2784  010616  020001                    CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2785  010620  001405                    BEQ     85$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2786  010622  004767  007756    84$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2787  010626  004767  011204            JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2788  010632  000007                    .WORD   7               ;ERROR TYPE CODE.
2789  010634                    85$:
2790  010634  005100                    COM     R0              ;COMPLEMENT CHECK WORD
2791  010636  005142                    COM     -(R2)           ;COMPLEMENT TEST DATA
2792  010640  012201                    MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2793  010642  020001                    CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2794  010644  001405                    BEQ     87$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2795  010646  004767  007732    86$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2796  010652  004767  011160            JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2797  010656  000007                    .WORD   7               ;ERROR TYPE CODE.
2798  010660                    87$:
2799  010660  005100                    COM     R0              ;COMPLEMENT CHECK WORD
2800  010662  005142                    COM     -(R2)           ;RESTORE DATA
2801  010664  012201                    MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2802  010666  020001                    CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2803  010670  001405                    BEQ     89$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2804  010672  004767  007706    88$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2805  010676  004767  011134            JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2806  010702  000007                    .WORD   7               ;ERROR TYPE CODE.
2807  010704                    89$:
2808  010704  012201                    MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2809  010706  020001                    CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2810  010710  001405                    BEQ     91$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2811  010712  004767  007666    90$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2812  010716  004767  011114            JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2813  010722  000007                    .WORD   7               ;ERROR TYPE CODE.
2814  010724                    91$:
2815  010724  005100                    COM     R0              ;COMPLEMENT CHECK WORD
2816  010726  005142                    COM     -(R2)           ;COMPLEMENT TEST DATA
2817  010730  012201                    MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2818  010732  020001                    CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2819  010734  001405                    BEQ     93$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2820  010736  004767  007642    92$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2821  010742  004767  011070            JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2822  010746  000007                    .WORD   7               ;ERROR TYPE CODE.
2823  010750                    93$:
2824  010750  005100                    COM     R0              ;COMPLEMENT CHECK WORD
2825  010752  005142                    COM     -(R2)           ;RESTORE DATA
2826  010754  012201                    MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
```

```
                                                    CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
                                                    BEQ     95$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
                                            94$:    JSR     PC,     $PRNT2  ;SET UP VALUES FOR ERROR PRINTING.
                                                    JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
                                                    .WORD   7               ;ERROR TYPE CODE.
                                            95$:
                                                    MOV     R0,     -(SP)   ;SAVE R0
                                                    MOV     R3,     R0      ;PUT R3 INTO R0
                                                    MOV     (SP)+,  R3      ;PUT SAVED R0 INTO R3
                                                    DEC     R4              ;DECREMENT 256. WORD COUNTER
                                                    BNE     22$             ;BRANCH IF MORE.
                                                    MOV     R0,     -(SP)   ;SAVE R0
                                                    MOV     R3,     R0      ;PUT R3 INTO R0
                                                    MOV     (SP)+,  R3      ;PUT SAVED R0 INTO R3
                                                    BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
                                                    BNE     21$             ;BRANCH IF MORE IN CURRENT BLOCK.
                                                    JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 21$.

                                            ;*****************************************************************
                                            ;*TEST 17       COMPLEMENT PARITY 3 XOR 9 TEST PATTERN.
                                            ;*****************************************************************
                                            TST17:
                                                    JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
                                                    .WORD   777             ;MINIMUM BLOCK SIZE OF 256. WORDS
                                                                            ;  REQUIRED FOR THIS TEST.
                                                    JMP     TST20           ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                                                                            ;  AVAILABLE FOR TEST.
                                                    MOV     #$TN-1,$TESTN   ;;SET TEST NUMBER IN  MAIL BOX
                                                    MOV     #-1,    R0      ;SET UP ALL ONES PATTERN
                                                    MOV     #401,   R3      ;SET UP PARITY "ALL ZEROS" PATTERN
                                                    JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
                                            1$:     JSR     PC,     W3X9    ;WRITE 256. WORD BLOCK WITH 3 XOR 9 PAT.
                                                    BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
                                                    BNE     1$              ;BRANCH IF MORE IN CURRENT BLOCK.
                                                    JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.

                                            ;*****************************************************************
                                            ;*  CHECK COMPLEMENT PARITY 3 XOR 9 PATTERN WRITTEN ABOVE.
                                            ;*****************************************************************
                                                    MOV     #-1,    R0      ;SET UP ALL ONES PATTERN
                                                    MOV     #401,   R3      ;SET UP PARITY "ALL ZEROS" PATTERN
                                                    JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
                                            11$:    MOV     #64.,   R4      ;SET 256. WORD COUNTER
                                            12$:
                                                    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,     $PRNT2  ;SET UP VALUES FOR ERROR PRINTING.
                                                    JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
                                                    .WORD   7               ;ERROR TYPE CODE.
                                            65$:
                                                    MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
                                                    CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
                                                    BEQ     67$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
                                            66$:    JSR     PC,     $PRNT2  ;SET UP VALUES FOR ERROR PRINTING.
                                                    JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
```

```
                                        .WORD   7               ;ERROR TYPE CODE.
                                67$:
                                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
                                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
                                        BEQ     69$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
                                68$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
                                        JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
                                        .WORD   7               ;ERROR TYPE CODE.
                                69$:
                                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
                                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
                                        BEQ     71$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
                                70$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
                                        JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
                                        .WORD   7               ;ERROR TYPE CODE.
                                71$:
                                        MOV     R0,     -(SP)   ;SAVE R0
                                        MOV     R3,     R0      ;PUT R3 INTO R0
                                        MOV     (SP)+,  R3      ;PUT SAVED R0 INTO R3
                                        DEC     R4              ;COUNT 256. WORDS
                                        BNE     12$             ;BRANCH IF MORE
                                        MOV     R0,     -(SP)   ;SAVE R0
                                        MOV     R3,     R0      ;PUT R3 INTO R0
                                        MOV     (SP)+,  R3      ;PUT SAVED R0 INTO R3
                                        BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
                                        BNE     11$             ;BRANCH IF MORE IN CURRENT BLOCK.
                                        JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 11$.

                                ;*****************************************************************
                                ;*  CHECK, COM, CHECK, COM, CHECK COMPLEMENTED PARITY 3 XOR 9 PATTERN.
                                ;*****************************************************************
                                        MOV     #-1,    R0      ;SET UP ALL ONES PATTERN
                                        MOV     #401,   R3      ;SET UP PARITY "ALL ZEROS" PATTERN
                                        JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
                                21$:    MOV     #64.,R4         ;SET 256. WORD COUNTER
                                22$:
                                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
                                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
                                        BEQ     73$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
                                72$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
                                        JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
                                        .WORD   7               ;ERROR TYPE CODE.
                                73$:
                                        COM     R0              ;COMPLEMENT CHECK WORD
                                        COM     -(R2)           ;COMPLEMENT TEST DATA
                                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
                                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
                                        BEQ     75$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
                                74$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
                                        JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
                                        .WORD   7               ;ERROR TYPE CODE.
                                75$:
                                        COM     R0              ;COMPLEMENT CHECK WORD
                                        COM     -(R2)           ;RESTORE DATA
                                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
                                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
```

```
2939  011340  001405                        BEQ     77$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2940  011342  004767  007236        76$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2941  011346  004767  010464                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2942  011352  000007                        .WORD   7               ;ERROR TYPE CODE.
2943  011354                        77$:
2944  011354  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2945  011356  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2946  011360  001405                        BEQ     79$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2947  011362  004767  007216        78$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2948  011366  004767  010444                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2949  011372  000007                        .WORD   7               ;ERROR TYPE CODE.
2950  011374                        79$:
2951  011374  005100                        COM     R0              ;COMPLEMENT CHECK WORD
2952  011376  005142                        COM     -(R2)           ;COMPLEMENT TEST DATA
2953  011400  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2954  011402  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2955  011404  001405                        BEQ     81$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2956  011406  004767  007172        80$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2957  011412  004767  010420                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2958  011416  000007                        .WORD   7               ;ERROR TYPE CODE.
2959  011420                        81$:
2960  011420  005100                        COM     R0              ;COMPLEMENT CHECK WORD
2961  011422  005142                        COM     -(R2)           ;RESTORE DATA
2962  011424  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2963  011426  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2964  011430  001405                        BEQ     83$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2965  011432  004767  007146        82$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2966  011436  004767  010374                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2967  011442  000007                        .WORD   7               ;ERROR TYPE CODE.
2968  011444                        83$:
2969  011444  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2970  011446  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2971  011450  001405                        BEQ     85$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2972  011452  004767  007126        84$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2973  011456  004767  010354                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2974  011462  000007                        .WORD   7               ;ERROR TYPE CODE.
2975  011464                        85$:
2976  011464  005100                        COM     R0              ;COMPLEMENT CHECK WORD
2977  011466  005142                        COM     -(R2)           ;COMPLEMENT TEST DATA
2978  011470  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2979  011472  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2980  011474  001405                        BEQ     87$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2981  011476  004767  007102        86$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2982  011502  004767  010330                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2983  011506  000007                        .WORD   7               ;ERROR TYPE CODE.
2984  011510                        87$:
2985  011510  005100                        COM     R0              ;COMPLEMENT CHECK WORD
2986  011512  005142                        COM     -(R2)           ;RESTORE DATA
2987  011514  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2988  011516  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2989  011520  001405                        BEQ     89$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2990  011522  004767  007056        88$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2991  011526  004767  010304                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2992  011532  000007                        .WORD   7               ;ERROR TYPE CODE.
2993  011534                        89$:
2994  011534  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
```

```
 2995  011536  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
 2996  011540  001405                        BEQ     91$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
 2997  011542  004767  007036        90$:    JSR     PC,     $PRNT2  ;SET UP VALUES FOR ERROR PRINTING.
 2998  011546  004767  010264                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
 2999  011552  000007                        .WORD   7               ;ERROR TYPE CODE.
 3000  011554                        91$:
 3001  011554  005100                        COM     R0              ;COMPLEMENT CHECK WORD
 3002  011556  005142                        COM     -(R2)           ;COMPLEMENT TEST DATA
 3003  011560  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
 3004  011562  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
 3005  011564  001405                        BEQ     93$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
 3006  011566  004767  007012        92$:    JSR     PC,     $PRNT2  ;SET UP VALUES FOR ERROR PRINTING.
 3007  011572  004767  010240                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
 3008  011576  000007                        .WORD   7               ;ERROR TYPE CODE.
 3009  011600                        93$:
 3010  011600  005100                        COM     R0              ;COMPLEMENT CHECK WORD
 3011  011602  005142                        COM     -(R2)           ;RESTORE DATA
 3012  011604  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
 3013  011606  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
 3014  011610  001405                        BEQ     95$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
 3015  011612  004767  006766        94$:    JSR     PC,     $PRNT2  ;SET UP VALUES FOR ERROR PRINTING.
 3016  011616  004767  010214                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
 3017  011622  000007                        .WORD   7               ;ERROR TYPE CODE.
 3018  011624                        95$:
 3019  011624  010046                        MOV     R0,     -(SP)   ;SAVE R0
 3020  011626  010300                        MOV     R3,     R0      ;PUT R3 INTO R0
 3021  011630  012603                        MOV     (SP)+,  R3      ;PUT SAVED R0 INTO R3
 3022  011632  005304                        DEC     R4              ;DECREMENT 256. WORD COUNTER
 3023  011634  001213                        BNE     22$             ;BRANCH IF MORE.
 3024  011636  010046                        MOV     R0,     -(SP)   ;SAVE R0
 3025  011640  010300                        MOV     R3,     R0      ;PUT R3 INTO R0
 3026  011642  012603                        MOV     (SP)+,  R3      ;PUT SAVED R0 INTO R3
 3027  011644  030502                        BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
 3028  011646  001204                        BNE     21$             ;BRANCH IF MORE IN CURRENT BLOCK.
 3029  011650  004767  003632                JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 21$.
```

```
3030                                    ;*********************************************************************
3031                                    ;*TEST 20       8 XOR 13 TEST PATTERN
3032                                    ;*********************************************************************
3033  011654                            TST20:
3034  011654  004567  007166                   JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
3035  011660  017777                           .WORD   17777           ;MINIMUM BLOCK SIZE OF 4096. WORDS
3036                                                                    ; REQUIRED FOR THIS TEST.
3037  011662  000167  000326                   JMP     TST21           ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
3038                                                                    ; AVAILABLE FOR TEST.
3039  011666  012767  000020  167312           MOV     #$TN-1,$TESTN   ;;SET TEST NUMBER IN  MAIL BOX
3040  011674  005000                    .8X13: CLR     R0
3041  011676  004467  003026                   JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
3042  011702  012704  000040            1$:    MOV     #32.,   R4      ;SET UP 4K LOOP COUNTER
3043  011706  005100                           COM     R0
3044  011710  012703  000200            2$:    MOV     #128.,  R3      ;EACH SMALL LOOP WRITES 128 WORDS
3045  011714  005100                           COM     R0
3046  011716  010022                    3$:    MOV     R0,     (R2)+   ;WRITE INTO MEMORY ADDRESSES
3047  011720  005303                           DEC     R3              ;DECREMENT WORD COUNT
3048  011722  001375                           BNE     3$
3049  011724  005304                           DEC     R4              ;DECREMENT 128. WORD COUNT
3050  011726  001370                           BNE     2$
3051  011730  030502                           BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
3052  011732  001353                           BNE     1$              ;BRANCH IF MORE IN CURRENT BLOCK.
3053  011734  004767  003546                   JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
3054
3055                                    ;*********************************************************************
3056                                    ;*  CHECK 8 XOR 13 TEST PATTERN WRITTEN ABOVE
3057                                    ;*********************************************************************
3058  011740  005000                           CLR     R0              ;SET UP CHECK WORD.
3059  011742  004467  002762                   JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
3060  011746  012704  000040            11$:   MOV     #32.,   R4      ;SET 4K WORD COUNTER
3061  011752  005100                           COM     R0              ;COMPLEMENT TEST WORD
3062  011754  012703  000100            12$:   MOV     #64.,   R3      ;SET 128 WORD COUNTER
3063  011760  005100                           COM     R0              ;COMPLEMENT TEST WORD
3064  011762                            13$:
3065  011762  012201                           MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
3066  011764  020001                           CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
3067  011766  001405                           BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
3068  011770  004767  006610            64$:   JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
3069  011774  004767  010036                   JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
3070  012000  000010                           .WORD   10              ;ERROR TYPE CODE.
3071  012002                            65$:
3072  012002  012201                           MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
3073  012004  020001                           CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
3074  012006  001405                           BEQ     67$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
3075  012010  004767  006570            66$:   JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
3076  012014  004767  010016                   JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
3077  012020  000010                           .WORD   10              ;ERROR TYPE CODE.
3078  012022                            67$:
3079  012022  005303                           DEC     R3              ;DECREMENT 128 WORD COUNTER
3080  012024  001356                           BNE     13$             ;BR IF MORE.
3081  012026  005304                           DEC     R4              ;DECREMENT 4096. WORD COUNTER
3082  012030  001351                           BNE     12$             ;BR IF MORE.
3083  012032  030502                           BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
3084  012034  001344                           BNE     11$             ;BRANCH IF MORE IN CURRENT BLOCK.
3085  012036  004767  003444                   JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 11$.
```

```
3086
3087                                    ;*********************************************************************
3088                                    ;*  COMPLEMENT 8 XOR 13 TEST PATTERN.
3089                                    ;*********************************************************************
3090  012042  016746  167536                   MOV     BLKMSK, -(SP)   ;SAVE BLOCK MASK TEMPORARILY.
3091  012046  012767  017777  167530           MOV     #MASK4K,BLKMSK  ;SET BLOCK MASK TO 4K.
3092  012054  004467  002650                   JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
3093  012060  012704  004000            21$:   MOV     #2048., R4      ;SET 4096. WORD COUNTER
3094  012064  005122                    22$:   COM     (R2)+           ;COMPLEMENT TEST PATTERN
3095  012066  005122                           COM     (R2)+
3096  012070  005304                           DEC     R4              ;COUNT 4K WORDS
3097  012072  001374                           BNE     22$             ;BR IF MORE.
3098  012074  030502                           BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
3099  012076  001370                           BNE     21$             ;BRANCH IF MORE IN CURRENT BLOCK.
3100  012100  004767  003402                   JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 21$.
3101  012104  012667  167474                   MOV     (SP)+,  BLKMSK  ;RESTORE BLOCK MASK.
3102
3103                                    ;*********************************************************************
3104                                    ;*  CHECK COMPLEMENTED 8 XOR 13 TEST PATTERN WRITTEN ABOVE.
3105                                    ;*********************************************************************
3106  012110  012700  177777                   MOV     #-1,    R0      ;SET UP CHECK WORD.
3107  012114  004467  002610                   JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
3108  012120  012704  000040            31$:   MOV     #32.,   R4      ;SET 4K WORD COUNTER
3109  012124  005100                           COM     R0              ;COMPLEMENT TEST WORD
3110  012126  012703  000100            32$:   MOV     #64.,   R3      ;SET 128 WORD COUNTER
3111  012132  005100                           COM     R0              ;COMPLEMENT TEST WORD
3112  012134                             33$:
3113  012134  012201                           MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
3114  012136  020001                           CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
3115  012140  001405                           BEQ     69$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
3116  012142  004767  006436            68$:   JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
3117  012146  004767  007664                   JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
3118  012152  000010                           .WORD   10              ;ERROR TYPE CODE.
3119  012154                             69$:
3120  012154  012201                           MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
3121  012156  020001                           CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
3122  012160  001405                           BEQ     71$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
3123  012162  004767  006416            70$:   JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
3124  012166  004767  007644                   JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
3125  012172  000010                           .WORD   10              ;ERROR TYPE CODE.
3126  012174                             71$:
3127  012174  005303                           DEC     R3              ;DECREMENT 128 WORD COUNTER
3128  012176  001356                           BNE     33$             ;BR IF MORE.
3129  012200  005304                           DEC     R4              ;DECREMENT 4096. WORD COUNTER
3130  012202  001351                           BNE     32$             ;BR IF MORE.
3131  012204  030502                           BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
3132  012206  001344                           BNE     31$             ;BRANCH IF MORE IN CURRENT BLOCK.
3133  012210  004767  003272                   JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 31$.
```

```
3134                                    ;*****************************************************************
3135                                    ;*TEST 21       WORSE CASE NOISE PARITY BYTE TESTING
3136                                    ;*  CHECK PARITY MEMORY WITH A SERIES OF BYTE PATTERNS
3137                                    ;*      1) FORCE WRONG PARITY IN EACH BYTE OF PARITY MEMORY
3138                                    ;*      2) READ IT BACK WITH ACTION ENABLE SET, MAKING SURE THAT A TRAP OCCURS
3139                                    ;*      3) WRITE GOOD PARITY AND MAKE SURE NO TRAP OCCURS WHEN IT IS READ
3140                                    ;*      4) MAKE SURE THE ERROR ADDRESS BITS (CSR BITS <11-5>) ARE CORRECT
3141                                    ;*****************************************************************
3142  012214                            TST21:
3143  012214  004567  006626                   JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
3144  012220  000000                           .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
3145  012222  012767  000021  166756          MOV     #$TN-1,$TESTN   ;;SET TEST NUMBER IN  MAIL BOX
3146  012230  005767  170016          WWPB0:  TST     MPRX            ;CHECK FOR ANY PARITY MEMORY.
3147  012234  001404                           BEQ     1$              ;BR IF NO PARITY MEMORY.
3148  012236  032777  000100  166672          BIT     #SW06,  @SWR    ;CHECK FORINHIBIT PARITY SWITCH.
3149  012244  001402                           BEQ     2$              ;BR IF NOT SET.
3150  012246  000167  000606          1$:     JMP     TST22           ;SKIP THIS TEST IF NO PARITY MEMORY PRESENT.
3151  012252  005000                  2$:     CLR     R0              ;ZERO TO BE PUT IN ALL MEMORY.
3152  012254  004767  004212                   JSR     PC,     SETCON  ;ROUTINE TO LOAD ALL MEMORY.
3153  012260  004467  002444                   JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
3154  012264  036767  167252  167244  WWPBYT: BIT     BITPT,  PMEMAP  ;CHECK IF CURRENT BANK HAS PARITY MEMORY.
3155  012272  001010                           BNE     2$              ;BR IF PARITY MEM.
3156  012274  036767  167244  167236          BIT     BITPT+2,PMEMAP+2 ;...HI 64K.
3157  012302  001004                           BNE     2$              ;BR IF PARITY MEM.
3158  012304  050502                           BIS     R5,     R2      ;POINT TO END OF BLOCK.
3159  012306  005202                           INC     R2              ;FIRST ADR OF NEXT BLOCK.
3160  012310  000167  000524                   JMP     WWPB5           ;BR TO FIND NEXT BLOCK.
3161  012314  004767  005500          2$:     JSR     PC,     SETAE   ;SET ACTION ENABLE (EVEN IF BANK0.)
3162  012320  004767  005530                   JSR     PC,     CKPMER  ;CHECK FOR ANY NON TRAP PARITY ERRORS.
3163  012324  020227  000114          WWPB1:  CMP     R2,     #114    ;CKECK IF POINTING TO PARITY ERROR VECTOR.
3164  012330  001004                           BNE     3$              ;BR IF NOT AT VECTOR.
3165  012332  062702  000004                   ADD     #4,     R2      ;SKIP PARITY VECTOR.
3166  012336  000167  000476                   JMP     WWPB5           ;CHECK FOR BLOCK END.
3167  012342  111201                  3$:     MOVB    (R2),   R1      ;CHECK IF BYTE STILL CLEARED.
3168  012344  001405                           BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
3169  012346  004767  006156          64$:    JSR     PC,     $PRNT   ;SET UP VALUES FOR ERROR PRINTING.
3170  012352  004767  007460                   JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
3171  012356  000011                           .WORD   11              ;ERROR TYPE CODE.
3172  012360                          65$:
3173  012360  105067  167172                   CLRB    OEFLG           ;CLEAR ODD/EVEN FLAG.
3174  012364  112700  000252                   MOVB    #252,   R0      ;SET UP DATA...EVEN, SETS PARITY BIT.
3175  012370  110012          WWPB2:  MOVB    R0,     (R2)    ;MOV DATA INTO TEST LOCATION.
3176  012372  016703  167220                   MOV     .MPRX,  R3      ;GET PARITY REGISTER TABLE POINTER.
3177  012376  052733  000005          10$:    BIS     #WWP+AE,@(R3)+  ;SET WRITE WRONG PARITY.
3178  012402  005713                           TST     (R3)            ;CHECK FOR TABLE TERMINATOR.
3179  012404  001374                           BNE     10$             ;BR IF MORE REGS IN TABLE.
3180                                  ;*  SET WRONG PARITY IN LOCATION UNDER TEST.
3181  012406  110012                           MOVB    R0,     (R2)    ;WRITE SAME DATA (EXCEPT PARITY) VIA DATOB.
3182  012410  016703  167202                   MOV     .MPRX,  R3      ;GET PARITY REG TABLE POINTER.
3183  012414  042733  000004          11$:    BIC     #WWP,   @(R3)+  ;CLEAR WRITE WRONG PARITY.
3184  012420  005713                           TST     (R3)            ;CHECK FOR TABLE TERMINATOR.
3185  012422  001374                           BNE     11$             ;BR IF MORE PARITY REGISTERS.
3186  012424  016737  167176  000114          MOV     .PBTRP, @#PARVEC ;SET UP VECTOR FOR EXPECTED TRAP.
3187                                  ;*  DETECT WRONG PARITY VIA DATIP; DATOB SHOULDN'T EXECUTE.
3188  012432  105412                           NEGB    (R2)            ;DATIP (DATOB AND COM PARITY BIT.)
3189                                  ;*  SHOULD HAVE TRAPPED TO PBTRP.
```

```
3190  012434  016737  167172  000114          MOV     .PESRV, @#PARVEC ;RESET VECTOR FOR UNEXPECTED TRAPS.
3191  012442  004767  006112          64$:     JSR     PC,     SPRNTO   ;SET UP VALUES FOR ERROR PRINTING.
3192  012446  004767  007364                   JSR     PC,     $ERROR   ;*** ERROR *** (GO TYPE A MESSAGE)
3193  012452  000012                           .WORD   12               ;ERROR TYPE CODE.
3194  012454  000557                           BR      WWPB4            ;SKIP TRAP SERVICE.
3195
3196                                  ;*  EXPECTED PARITY MEMORY TRAPS COME HERE.
3197  012456  016737  167150  000114  PBTRP:  MOV     .PESRV, @#PARVEC ;RESET PARITY VECTOR FOR UNEXPECTED TRAPS.
3198  012464  022626                   CMP     (SP)+,  (SP)+    ;RESET THE STACK POINTER AFTER TRAP.
3199  012466  016703  167122                   MOV     .MPRO,  R3       ;GET PARITY REG AND MAP TABLE POINTER.
3200  012472  032713  000001          21$:     BIT     #BIT0,  (R3)     ;CHECK IF THIS REGISTER EXISTS.
3201  012476  001003                           BNE     22$              ;BR IF IT DOESN'T EXIST.
3202  012500  017301  000000                   MOV     @(R3),  R1       ;GET THE CONTENTS.
3203  012504  100413                           BMI     23$              ;BR IF ERROR FLAG SET.
3204  012506  062703  000006          22$:     ADD     #6,     R3       ;MOVE POINTER TO NEXT REG.
3205  012512  020367  167100                   CMP     R3,     .MPRX    ;CHECK FOR END OF TABLE.
3206  012516  103765                           BLO     21$              ;BR IF MORE REGISTERS.
3207  012520  004767  006034          64$:     JSR     PC,     SPRNTO   ;SET UP VALUES FOR ERROR PRINTING.
3208  012524  004767  007306                   JSR     PC,     $ERROR   ;*** ERROR *** (GO TYPE A MESSAGE)
3209  012530  000013                           .WORD   13               ;ERROR TYPE CODE.
3210  012532  000530                           BR      WWPB4            ;EXIT AFTER ERROR.
3211  012534  036763  167002  000002  23$:     BIT     BITPT,  2(R3)    ;CHECK THE MAP FOR THIS REGISTER.
3212  012542  001011                           BNE     24$              ;BR IF THIS REGISTER CONTROLS THIS BANK.
3213  012544  036763  166774  000004          BIT     BITPT+2,4(R3)    ;CHECK THE HI 64K.
3214  012552  001005                           BNE     24$              ;BR IF THIS REGISTER CONTROLS THIS BANK.
3215  012554  004767  005774          65$:     JSR     PC,     SPRNTP   ;SET UP VALUES FOR ERROR PRINTING.
3216  012560  004767  007252                   JSR     PC,     $ERROR   ;*** ERROR *** (GO TYPE A MESSAGE)
3217  012564  000014                           .WORD   14               ;ERROR TYPE CODE.
3218  012566                          24$:
3219  012566  010046                           MOV     R0,-(SP)         ;;PUSH R0 ON STACK
3220  012570  010200                           MOV     R2,     R0       ;GET THE ADDRESS POINTER.
3221  012572  042700  003777                   BIC     #3777,  R0       ;CLEAR LOW ADDRESS BITS.
3222  012576  000300                           SWAB    R0               ;SHIFT 6 PLACES RIGHT.
3223  012600  006300                           ASL     R0
3224  012602  006300                           ASL     R0
3225  012604  005767  165776                   TST     MMAVA            ;CHECK FOR MEM MGMT.
3226  012610  001404                           BEQ     25$              ;BR IF NO MEM MGMT.
3227  012612  042700  177600                   BIC     #177600,R0       ;CLEAR BANK BITS
3228  012616  063700  172344                   ADD     @#KIPAR2,R0      ;ADD MEM MGMT OFFSET.
3229  012622  052700  100001          25$:     BIS     #BIT15+BIT0,R0   ;SET ERROR AND AE BIT IN CHECK WORD.
3230  012626  046700  166662                   BIC     RESRVD, R0       ;CLEAR PARITY REG BITS RESERVED FOR FUTURE.
3231  012632  046701  166656                   BIC     RESRVD, R1       ;CLEAR PARITY REG BITS RESERVED FOR FUTURE.
3232                                  ;NOTE: THE ABOVE INSTRUCTION (2 WORDS) CAN BE NOP'ED FOR UNMIXED MEMORY TYPES.
3233  012636  020001                           CMP     R0,     R1       ;COMPARE THE CHECK WORD WITH THE DATA READ.
3234  012640  001405                           BEQ     67$              ;BRANCH OVER ERROR CALL IF GOOD DATA.
3235  012642  004767  005706          66$:     JSR     PC,     SPRNTP   ;SET UP VALUES FOR ERROR PRINTING.
3236  012646  004767  007164                   JSR     PC,     $ERROR   ;*** ERROR *** (GO TYPE A MESSAGE)
3237  012652  000015                           .WORD   15               ;ERROR TYPE CODE.
3238  012654                          67$:
3239  012654  005073  000000                   CLR     @(R3)            ;CLEAR REG INCLUDING ACTION ENABLE.
3240  012660  010346                           MOV     R3,-(SP)         ;;PUSH R3 ON STACK
3241  012662  062703  000006          26$:     ADD     #6,     R3       ;UPDATE POINTER TO NEXT PARITY REG + MAP.
3242  012666  020367  166724                   CMP     R3,     .MPRX    ;CHECK FOR END OF TABLE.
3243  012672  101014                           BHI     WWPB3            ;BR IF END OF TABLE REACHED.
3244  012674  032713  000001                   BIT     #BIT0,  (R3)     ;CHECK IF NEXT REG EXISTS.
3245  012700  001370                           BNE     26$              ;BR IF THIS PARITY REG DOESN'T EXIST.
```

```
3246  012702  017301  000000                  MOV     @(R3),  R1      ;SAVE AND CHECK FOR ERROR FLAG.
3247  012706  100365                          BPL     26$             ;BR IF NO ERROR FLAG.
3248  012710  004767  005640          68$:    JSR     PC,     SPRNTP  ;SET UP VALUES FOR ERROR PRINTING.
3249  012714  004767  007116                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
3250  012720  000016                          .WORD   16              ;ERROR TYPE CODE.
3251  012722  000757                          BR      26$             ;BR AFTER ERROR.
3252  012724  111204                  WWPB3:  MOVB    (R2),   R4      ;GET THE DATA FOR CHECKING.
3253                                  ;* READING THE DATA VIA DATI TO CHECK IT SHOULD CAUSE PARITY ERROR, BUT
3254                                  ;*  ACTION ENABLE IS NOT SET IN CONTROLING REG, SO NO TRAP SHOULD OCCURE.
3255  012726  111212                          MOVB    (R2),   (R2)    ;RESTORE RIGHT PARITY
3256                                  ;NOTE:  THE ABOVE INSTRUCTION CAN BE NOP'ED FOR PROCESSORS
3257                                  ;       WHICH DO ONLY DATOB TO DESTINATION OF MOVB INSTRUCTIONS.
3258  012730  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
3259  012732  017301  000000                  MOV     @(R3),  R1      ;READ THE PARITY REGISTER TO CHECK IT AGAIN.
3260  012736  046701  166552                  BIC     RESRVD, R1      ;CLEAR PARITY REG BITS RESERVED FOR FUTURE.
3261                                  ;NOTE: THE ABOVE INSTRUCTION (2 WORDS) CAN BE NOP'ED FOR UNMIXED MEMORY TYPES.
3262  012742  042700  000001                  BIC     #AE,    R0      ;CLEAR THE ACTION ENABLE BIT IN TEST DATA.
3263  012746  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
3264  012750  001405                          BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
3265  012752  004767  005576          64$:    JSR     PC,     SPRNTP  ;SET UP VALUES FOR ERROR PRINTING.
3266  012756  004767  007054                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
3267  012762  000015                          .WORD   15              ;ERROR TYPE CODE.
3268  012764                          65$:
3269  012764  012773  000001  000000          MOV     #1,     @(R3)   ;CLEAR ALL BUT ACTION ENABLE.
3270  012772  010401                          MOV     R4,     R1      ;GET DATA READ FROM MEMORY FOR TESTING.
3271  012774  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
3272  012776  120001                          CMPB    R0,     R1      ;CHECK THE DATA.
3273  013000  001405                          BEQ     67$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
3274  013002  004767  005552          66$:    JSR     PC,     SPRNTO  ;SET UP VALUES FOR ERROR PRINTING.
3275  013006  004767  007024                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
3276  013012  000017                          .WORD   17              ;ERROR TYPE CODE.
3277  013014                          67$:
3278  013014  110012                  WWPB4:  MOVB    R0,     (R2)    ;RESTORE DATA.
3279  013016  105712                          TSTB    (R2)            ;DO A DATI TO BE SURE RIGHT PARITY.
3280  013020  012700  000253                  MOV     #253,   R0      ;SET ODD PARITY DATA.
3281  013024  105167  166526                  COMB    OEFLG           ;CHECK IF DONE BOTH ODD AND EVEN PARITY.
3282  013030  100002                          BPL     27$             ;BR IF DONE BOTH EVEN AND ODD.
3283  013032  000167  177332                  JMP     WWPB2           ;LOOP BACK AND DO ODD(PARITY BIT CLR).
3284  013036  005202                  27$:    INC     R2              ;MOVE POINTER TO NEXT MEMORY BYTE.
3285  013040  030502                  WWPB5:  BIT     R5,     R2      ;CHECK FOR END OF BLOCK.
3286  013042  001402                          BEQ     30$             ;BR IF END OF BLOCK FOUND.
3287  013044  000167  177254                  JMP     WWPB1           ;LOOP BACK TO TEST NEXT BYTE.
3288  013050  004767  002432          30$:    JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO WWPBYT
3289  013054  004767  004674                  JSR     PC,     MAMF    ;GO RESET PARITY REGISTERS.
```

```
3290                                    ;*****************************************************************
3291                                    ;*TEST 22       RANDOM DATA  TESTING THRU PROGRAM CODE RELOCATION.
3292                                    ;*****************************************************************
3293  013060                            TST22:
3294  013060  004567  005762                   JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
3295  013064  000000                            .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
3296  013066  012767  000022  166112           MOV     #$TN-1,$TESTN   ;;SET TEST NUMBER IN  MAIL BOX
3297  013074  010703                    RANTST: MOV     PC,     R3      ;GET CURRENT PROGRAM COUNTER.
3298  013076  042703  007777                   BIC     #7777,  R3      ;POINT TO BEGINNING OF CURRENT 2K BLOCK.
3299  013102  004467  001622                   JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
3300  013106  010246                    1$:     MOV     R2,     -(SP)   ;SAVE MEMORY POINTER.
3301  013110  010346                            MOV     R3,     -(SP)   ;SAVE "DATA" POINTER.
3302  013112  012322                    2$:     MOV     (R3)+,  (R2)+   ;MOV CODE INTO TEST MEMORY.
3303  013114  032703  007777                   BIT     #7777,  R3      ;CHECK FOR END OF "DATA TABLE"
3304  013120  001002                            BNE     3$              ;BRANCH IF MORE
3305  013122  162703  010000                   SUB     #10000, R3      ;RESET POINTER TO START OF "RANDOM DATA"
3306  013126  030502                    3$:     BIT     R5,     R2      ;CHECK FOR END OF BLOCK
3307  013130  001370                            BNE     2$              ;BRANCH IF MORE.
3308  013132  012603                            MOV     (SP)+,  R3      ;RESET "DATA" POINTER.
3309  013134  012602                            MOV     (SP)+,  R2      ;RESET MEMORY POINTER.
3310  013136  012300                    4$:     MOV     (R3)+,  R0      ;GET S/B DATA.
3311  013140  012201                            MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
3312  013142  020001                            CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
3313  013144  001405                            BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
3314  013146  004767  005432          64$:     JSR     PC,     $PRNT2  ;SET UP VALUES FOR ERROR PRINTING.
3315  013152  004767  006660                   JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
3316  013156  000020                            .WORD   20              ;ERROR TYPE CODE.
3317  013160                            65$:
3318  013160  032703  007777                   BIT     #7777,  R3      ;CHECK FOR END OF "DATA TABLE"
3319  013164  001002                            BNE     5$              ;BR IF MORE.
3320  013166  162703  010000                   SUB     #10000, R3      ;RESET POINTER TO TOP OF "DATA TABLE".
3321  013172                            5$:
3322  013172  030502                            BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
3323  013174  001360                            BNE     4$              ;BRANCH IF MORE IN CURRENT BLOCK.
3324  013176  004767  002304                   JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
```

```
                                .SBTTL SECTION 3:       INSTRUCTION EXECUTION TESTS.
                                ;*****************************************************************
                                ;*TEST 23       EXECUTE DATI, DATO THRU MEMORY.
                                ;*  EXECUTES THE INSTRUCTION 'MOV R4,(R2)' THROUGHOUT MEMORY.
                                ;*  AN 'RTS R5' (CODE 205) IS PLACED AFTER THE 'MOV' INSTRUCTION TO RETURN
                                ;*  CONTROL TO THE MAIN PROGRAM FOR INSTRUCTION EXECUTION CHECKOUT.
                                ;*  THIS IS AN EXAMPLE OF WHAT THIS TEST DOES IN RELATION TO MEMORY:
                                ;*
                                ;*                MEMORY        INSTRUCTION     CONTENTS OF MEMORY LOCATION
                                ;*               LOCATION       PLACED THERE    AFTER INSTRUCTION EXECUTION
                                ;*
                                ;*   1ST PASS /  40000          010412          000205
                                ;*  THRU TEST /  40002          000205          000205
                                ;*
                                ;*   2ND PASS /  40002          010412          000205
                                ;*  THRU TEST /  40004          000205          000205
                                ;*
                                ;*      ETC., ETC., ETC.
                                ;*
                                ;*  R0 = DATA WRITTEN ON TOP OF IUT BY THE IUT (SHOULD BE).
                                ;*  R1 = DATA READ FROM MEMORY (WAS).
                                ;*  R2 = ADDRESS OF IUT/DATA.
                                ;*  R3 = INSTRUCTION UNDER TEST (IUT).
                                ;*  R4 = RTS R5 (CODE 205).
                                ;*  R5 = BLOCK BOUNDRY BIT MASK.
                                ;*****************************************************************
013202                          TST23:
013202  004567  005640                  JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
013206  000003                  .WORD   3                       ;MINIMUM BLOCK SIZE OF 2 WORDS
                                                                ; REQUIRED FOR THIS TEST.
013210  000167  000064                  JMP     TST24           ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                                                                ; AVAILABLE FOR TEST.
013214  012767  000023  165764          MOV     #$TN-1,$TESTN   ;;SET TEST NUMBER IN  MAIL BOX
013222  012703  010412          DIDO:   MOV     #010412,R3      ;GET 'MOV R4,(R2)' INSTRUCTION (IUT).
013226  012704  000205                  MOV     #205,   R4      ;GET 'RTS R5'
013232  010400                          MOV     R4,     R0      ;SET UP S/B DATA AFTER EXECUTION.
013234  004467  001470                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
013240  010322                  1$:     MOV     R3,     (R2)+   ;PUT IUT INTO FIRST LOC OF BLOCK.
013242  010412                  2$:     MOV     R4,     (R2)    ;PUT 'RTS R5' FOLLOWING IUT.
013244  004542                          JSR     R5,     -(R2)   ;GO EXECUTE THE IUT.
013246  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM THE MEM ADR UNDER TEST.
013250  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
013252  001405                          BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
013254  004767  005320          64$:    JSR     PC,     $PRNT3  ;SET UP VALUES FOR ERROR PRINTING.
013260  004767  006552                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
013264  000021                  .WORD   21                      ;ERROR TYPE CODE.
013266                          65$:
013266  010322                          MOV     R3,     (R2)+   ;PUT THE IUT INTO THE NEXT LOCATION.
013270  030502                          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
013272  001363                          BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
013274  004767  002206                  JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
```

```
                                ;*****************************************************************
                                ;*TEST 24       EXECUTE DATI, DATOB (LOW BYTE) THRU MEMORY.
                                ;*  EXECUTES THE INSTRUCTION 'MOVB R4,(R2)' THROUGHOUT MEMORY.
                                ;*  AN 'RTS R5' (CODE 205) IS PLACED AFTER THE 'MOVB' INSTRUCTION TO RETURN
                                ;*  CONTROL TO THE MAIN PROGRAM FOR INSTRUCTION EXECUTION CHECKOUT.
                                ;*  THIS IS AN EXAMPLE OF WHAT THIS TEST DOES IN RELATION TO MEMORY:
                                ;*
                                ;*                MEMORY        INSTRUCTION     CONTENTS OF MEMORY LOCATION
                                ;*               LOCATION       PLACED THERE    AFTER INSTRUCTION EXECUTION
                                ;*
                                ;*   1ST PASS /  40000          110412          110605
                                ;*  THRU TEST /  40002          000205          000205
                                ;*
                                ;*   2ND PASS /  40002          110412          110605
                                ;*  THRU TEST /  40004          000205          000205
                                ;*
                                ;*     ETC., ETC., ETC.
                                ;*
                                ;*  R0 = DATA WRITTEN ON TOP OF IUT BY THE IUT (SHOULD BE).
                                ;*  R1 = DATA READ FROM MEMORY (WAS).
                                ;*  R2 = ADDRESS OF IUT/DATA.
                                ;*  R3 = INSTRUCTION UNDER TEST (IUT).
                                ;*  R4 = RTS R5 (CODE 205).
                                ;*  R5 = BLOCK BOUNDRY BIT MASK.
                                ;*****************************************************************
013300                          TST24:
013300  004567  005542                JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
013304  000003                        .WORD   3               ;MINIMUM BLOCK SIZE OF 2 WORDS
                                                                ; REQUIRED FOR THIS TEST.
013306  000167  000066                JMP     TST25           ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                                                                ; AVAILABLE FOR TEST.
013312  012767  000024  165666        MOV     #$TN-1,$TESTN   ;;SET TEST NUMBER IN  MAIL BOX
013320  012703  110412        DIDBL:  MOV     #110412,R3      ;GET 'MOVB R4,(R2)' INSTRUCTION (IUT).
013324  012704  000205                MOV     #205,   R4      ;GET 'RTS R5'
013330  012700  110605                MOV     #110605,R0      ;SET UP S/B DATA AFTER EXECUTION.
013334  004467  001370                JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
013340  010322                1$:     MOV     R3,     (R2)+   ;PUT IUT INTO FIRST LOC OF BLOCK.
013342  010412                2$:     MOV     R4,     (R2)    ;PUT 'RTS R5' FOLLOWING IUT.
013344  004542                        JSR     R5,     -(R2)   ;GO EXECUTE THE IUT.
013346  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM THE MEM ADR UNDER TEST.
013350  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
013352  001405                        BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
013354  004767  005220        64$:    JSR     PC,     $PRNT3  ;SET UP VALUES FOR ERROR PRINTING.
013360  004767  006452                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
013364  000021                        .WORD   21              ;ERROR TYPE CODE.
013366                          65$:
013366  010322                        MOV     R3,     (R2)+   ;PUT THE IUT INTO THE NEXT LOCATION.
013370  030502                        BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
013372  001363                        BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
013374  004767  002106                JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
```

```
                        ;*****************************************************************
                        ;*TEST 25       EXECUTE DATI, DATOB (HIGH BYTE) THRU MEMORY.
                        ;*  EXECUTES THE INSTRUCTION 'MOVB R3,-(R2)' THROUGHOUT MEMORY.
                        ;*  AN 'RTS R5' (CODE 205) IS PLACED AFTER THE 'MOVB' INSTRUCTION TO RETURN
                        ;*  CONTROL TO THE MAIN PROGRAM FOR INSTRUCTION EXECUTION CHECKOUT.
                        ;*  THIS IS AN EXAMPLE OF WHAT THIS TEST DOES IN RELATION TO MEMORY:
                        ;*
                        ;*                MEMORY        INSTRUCTION     CONTENTS OF MEMORY LOCATION
                        ;*               LOCATION      PLACED THERE    AFTER INSTRUCTION EXECUTION
                        ;*
                        ;*   1ST PASS /  40000          110342          161342
                        ;*  THRU TEST /  40002          000205          000205
                        ;*
                        ;*   2ND PASS /  40002          110342          161342
                        ;*  THRU TEST /  40004          000205          000205
                        ;*
                        ;*     ETC., ETC., ETC.
                        ;*
                        ;*  R0 = DATA WRITTEN ON TOP OF IUT BY THE IUT (SHOULD BE).
                        ;*  R1 = DATA READ FROM MEMORY (WAS).
                        ;*  R2 = ADDRESS OF IUT/DATA.
                        ;*  R3 = INSTRUCTION UNDER TEST (IUT).
                        ;*  R4 = RTS R5 (CODE 205).
                        ;*  R5 = BLOCK BOUNDRY BIT MASK.
                        ;*****************************************************************
013400                  TST25:
013400  004567  005442          JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
013404  000003          .WORD   3               ;MINIMUM BLOCK SIZE OF 2 WORDS
                                                ;  REQUIRED FOR THIS TEST.
013406  000167  000072          JMP     TST26           ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                                                ;  AVAILABLE FOR TEST.
013412  012767  000025  165566  MOV     #$TN-1,$TESTN   ;;SET TEST NUMBER IN  MAIL BOX
013420  012703  110342  DIDBH:  MOV     #110342,R3      ;GET 'MOVB R3,-(R2)' INSTRUCTION (IUT).
013424  012704  000205          MOV     #205,   R4      ;GET 'RTS R5'
013430  012700  161342          MOV     #161342,R0      ;SET UP S/B DATA AFTER EXECUTION.
013434  004467  001270          JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
013440  010322          1$:     MOV     R3,     (R2)+   ;PUT IUT INTO FIRST LOC OF BLOCK.
013442  010412          2$:     MOV     R4,     (R2)    ;PUT 'RTS R5' FOLLOWING IUT.
013444  004562  177776          JSR     R5,     -2(R2)  ;GO EXECUTE THE IUT.
013450  005302                  DEC     R2              ;ADJUST R2 TO POINT TO MAUT.
013452  012201                  MOV     (R2)+,  R1      ;GET THE DATA FROM THE MEM ADR UNDER TEST.
013454  020001                  CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
013456  001405                  BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
013460  004767  005114  64$:    JSR     PC,     $PRNT3  ;SET UP VALUES FOR ERROR PRINTING.
013464  004767  006346          JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
013470  000021          .WORD   21              ;ERROR TYPE CODE.
013472                  65$:
013472  010322                  MOV     R3,     (R2)+   ;PUT THE IUT INTO THE NEXT LOCATION.
013474  030502                  BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
013476  001361                  BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
013500  004767  002002          JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
```

```
                                ;*****************************************************************
                                ;*TEST 26       EXECUTE DATI, DATIP, DATO THRU MEMORY.
                                ;*  EXECUTES THE INSTRUCTION 'NEG (R2)' THROUGHOUT MEMORY.
                                ;*  AN 'RTS R5' (CODE 205) IS PLACED AFTER THE 'NEG' INSTRUCTION TO RETURN
                                ;*  CONTROL TO THE MAIN PROGRAM FOR INSTRUCTION EXECUTION CHECKOUT.
                                ;*  THIS IS AN EXAMPLE OF WHAT THIS TEST DOES IN RELATION TO MEMORY:
                                ;*
                                ;*                MEMORY        INSTRUCTION     CONTENTS OF MEMORY LOCATION
                                ;*               LOCATION      PLACED THERE    AFTER INSTRUCTION EXECUTION
                                ;*
                                ;*   1ST PASS /  40000          005412          172366
                                ;*  THRU TEST /  40002          000205          000205
                                ;*
                                ;*   2ND PASS /  40002          005412          172366
                                ;*  THRU TEST /  40004          000205          000205
                                ;*
                                ;*      ETC., ETC., ETC.
                                ;*
                                ;*  R0 = DATA WRITTEN ON TOP OF IUT BY THE IUT (SHOULD BE).
                                ;*  R1 = DATA READ FROM MEMORY (WAS).
                                ;*  R2 = ADDRESS OF IUT/DATA.
                                ;*  R3 = INSTRUCTION UNDER TEST (IUT).
                                ;*  R4 = RTS R5 (CODE 205).
                                ;*  R5 = BLOCK BOUNDRY BIT MASK.
                                ;*****************************************************************
013504                          TST26:
013504  004567  005336                  JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
013510  000003                  .WORD   3                       ;MINIMUM BLOCK SIZE OF 2 WORDS
                                                                ; REQUIRED FOR THIS TEST.
013512  000167  000066          JMP     TST27                   ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                                                                ; AVAILABLE FOR TEST.
013516  012767  000026  165462  MOV     #$TN-1,$TESTN           ;;SET TEST NUMBER IN  MAIL BOX
013524  012703  005412  DIPDO:  MOV     #005412,R3              ;GET 'NEG (R2)' INSTRUCTION (IUT).
013530  012704  000205          MOV     #205,   R4              ;GET 'RTS R5'
013534  012700  172366          MOV     #172366,R0              ;SET UP S/B DATA AFTER EXECUTION.
013540  004467  001164          JSR     R4,     INITMM          ;INITIALIZE THE MEMORY ADDRESS POINTERS.
013544  010322  1$:     MOV     R3,     (R2)+                   ;PUT IUT INTO FIRST LOC OF BLOCK.
013546  010412  2$:     MOV     R4,     (R2)                    ;PUT 'RTS R5' FOLLOWING IUT.
013550  004542          JSR     R5,     -(R2)                   ;GO EXECUTE THE IUT.
013552  012201          MOV     (R2)+,  R1                      ;GET THE DATA FROM THE MEM ADR UNDER TEST.
013554  020001          CMP     R0,     R1                      ;COMPARE THE CHECK WORD WITH THE DATA READ.
013556  001405          BEQ     65$                             ;BRANCH OVER ERROR CALL IF GOOD DATA.
013560  004767  005014  64$:    JSR     PC,     $PRNT3          ;SET UP VALUES FOR ERROR PRINTING.
013564  004767  006246          JSR     PC,     $ERROR          ;*** ERROR *** (GO TYPE A MESSAGE)
013570  000021          .WORD   21                              ;ERROR TYPE CODE.
013572                  65$:
013572  010322          MOV     R3,     (R2)+                   ;PUT THE IUT INTO THE NEXT LOCATION.
013574  030502          BIT     R5,     R2                      ;CHECK FOR END OF A BLOCK.
013576  001363          BNE     2$                              ;BRANCH IF MORE IN CURRENT BLOCK.
013600  004767  001702          JSR     PC,     MMUP            ;FIND NEXT BLOCK AND LOOP TO 1$.
```

```
                                ;*****************************************************************
                                ;*TEST 27       EXECUTE DATI, DATI, DATIP, DATOB (LOW BYTE) THRU MEMORY.
                                ;*  EXECUTES THE INSTRUCTION 'BICB (R2)+,-(R2)' THROUGHOUT MEMORY.
                                ;*  AN 'RTS R5' (CODE 205) IS PLACED AFTER THE 'BICB' INSTRUCTION TO RETURN
                                ;*  CONTROL TO THE MAIN PROGRAM FOR INSTRUCTION EXECUTION CHECKOUT.
                                ;*  THIS IS AN EXAMPLE OF WHAT THIS TEST DOES IN RELATION TO MEMORY:
                                ;*
                                ;*                MEMORY        INSTRUCTION      CONTENTS OF MEMORY LOCATION
                                ;*               LOCATION       PLACED THERE     AFTER INSTRUCTION EXECUTION
                                ;*
                                ;*   1ST PASS /  40000           142242           142000
                                ;*  THRU TEST /  40002           000205           000205
                                ;*
                                ;*   2ND PASS /  40002           142242           142000
                                ;*  THRU TEST /  40004           000205           000205
                                ;*
                                ;*      ETC., ETC., ETC.
                                ;*
                                ;*  R0 = DATA WRITTEN ON TOP OF IUT BY THE IUT (SHOULD BE).
                                ;*  R1 = DATA READ FROM MEMORY (WAS).
                                ;*  R2 = ADDRESS OF IUT/DATA.
                                ;*  R3 = INSTRUCTION UNDER TEST (IUT).
                                ;*  R4 = RTS R5 (CODE 205).
                                ;*  R5 = BLOCK BOUNDRY BIT MASK.
                                ;*****************************************************************
013604                          TST27:
013604  004567  005236                  JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
013610  000003                  .WORD   3                       ;MINIMUM BLOCK SIZE OF 2 WORDS
                                                                ; REQUIRED FOR THIS TEST.
013612  000167  000066          JMP     TST30                   ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                                                                ; AVAILABLE FOR TEST.
013616  012767  000027  165362  DPDBL:  MOV     #$TN-1,$TESTN   ;;SET TEST NUMBER IN  MAIL BOX
013624  012703  142242          MOV     #142242,R3              ;GET BICB (R2)+,-(R2)' INSTRUCTION (IUT).
013630  012704  000205          MOV     #205,   R4              ;GET 'RTS R5'
013634  012700  142000          MOV     #142000,R0              ;SET UP S/B DATA AFTER EXECUTION.
013640  004467  001064          JSR     R4,     INITMM          ;INITIALIZE THE MEMORY ADDRESS POINTERS.
013644  010322                  1$:     MOV     R3,     (R2)+   ;PUT IUT INTO FIRST LOC OF BLOCK.
013646  010412                  2$:     MOV     R4,     (R2)    ;PUT 'RTS R5' FOLLOWING IUT.
013650  004542                  JSR     R5,     -(R2)           ;GO EXECUTE THE IUT.
013652  012201                  MOV     (R2)+,  R1              ;GET THE DATA FROM THE MEM ADR UNDER TEST.
013654  020001                  CMP     R0,     R1              ;COMPARE THE CHECK WORD WITH THE DATA READ.
013656  001405                  BEQ     65$                     ;BRANCH OVER ERROR CALL IF GOOD DATA.
013660  004767  004714          64$:    JSR     PC,     $PRNT3  ;SET UP VALUES FOR ERROR PRINTING.
013664  004767  006146          JSR     PC,     $ERROR          ;*** ERROR *** (GO TYPE A MESSAGE)
013670  000021                  .WORD   21                      ;ERROR TYPE CODE.
013672                          65$:
013672  010322                  MOV     R3,     (R2)+           ;PUT THE IUT INTO THE NEXT LOCATION.
013674  030502                  BIT     R5,     R2              ;CHECK FOR END OF A BLOCK.
013676  001363                  BNE     2$                      ;BRANCH IF MORE IN CURRENT BLOCK.
013700  004767  001602          JSR     PC,     MMUP            ;FIND NEXT BLOCK AND LOOP TO 1$.
```

```
                                ;*****************************************************************
                                ;*TEST 30       EXECUTE DATI, DATI, DATIP, DATOB (HIGH BYTE) THRU MEMORY.
                                ;*  EXECUTES THE INSTRUCTION 'BISB (R2)+,(R2)' THROUGHOUT MEMORY.
                                ;*  AN 'RTS R5' (CODE 205) IS PLACED AFTER THE 'BISB' INSTRUCTION TO RETURN
                                ;*  CONTROL TO THE MAIN PROGRAM FOR INSTRUCTION EXECUTION CHECKOUT.
                                ;*  THIS IS AN EXAMPLE OF WHAT THIS TEST DOES IN RELATION TO MEMORY:
                                ;*
                                ;*                MEMORY          INSTRUCTION     CONTENTS OF MEMORY LOCATION
                                ;*               LOCATION         PLACED THERE    AFTER INSTRUCTION EXECUTION
                                ;*
                                ;*   1ST PASS /  40000            152212          157212
                                ;*  THRU TEST /  40002            000205          000205
                                ;*
                                ;*   2ND PASS /  40002            152212          157212
                                ;*  THRU TEST /  40004            000205          000205
                                ;*
                                ;*     ETC., ETC., ETC.
                                ;*
                                ;*  R0 = DATA WRITTEN ON TOP OF IUT BY THE IUT (SHOULD BE).
                                ;*  R1 = DATA READ FROM MEMORY (WAS).
                                ;*  R2 = ADDRESS OF IUT/DATA.
                                ;*  R3 = INSTRUCTION UNDER TEST (IUT).
                                ;*  R4 = RTS R5 (CODE 205).
                                ;*  R5 = BLOCK BOUNDRY BIT MASK.
                                ;*****************************************************************
013704                          TST30:
013704  004567  005136                JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
013710  000003                    .WORD   3               ;MINIMUM BLOCK SIZE OF 2 WORDS
                                                        ; REQUIRED FOR THIS TEST.
013712  000167  000070            JMP     TST31           ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                                                        ; AVAILABLE FOR TEST.
013716  012767  000030  165262    MOV     #$TN-1,$TESTN   ;;SET TEST NUMBER IN  MAIL BOX
013724  012703  152212    DPDBH:  MOV     #152212,R3      ;GET 'BISB (R2)+,(R2)' INSTRUCTION (IUT).
013730  012704  000205            MOV     #205,   R4      ;GET 'RTS R5'
013734  012700  157212            MOV     #157212,R0      ;SET UP S/B DATA AFTER EXECUTION.
013740  004467  000764            JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
013744  010322            1$:     MOV     R3,     (R2)+   ;PUT IUT INTO FIRST LOC OF BLOCK.
013746  010412            2$:     MOV     R4,     (R2)    ;PUT 'RTS R5' FOLLOWING IUT.
013750  004542                    JSR     R5,     -(R2)   ;GO EXECUTE THE IUT.
013752  005302                    DEC     R2              ;RESET R2 TO POINT TO IUT.
013754  012201                    MOV     (R2)+,  R1      ;GET THE DATA FROM THE MEM ADR UNDER TEST.
013756  020001                    CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
013760  001405                    BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
013762  004767  004612    64$:    JSR     PC,     $PRNT3  ;SET UP VALUES FOR ERROR PRINTING.
013766  004767  006044            JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
013772  000021                    .WORD   21              ;ERROR TYPE CODE.
013774                    65$:
013774  010322                    MOV     R3,     (R2)+   ;PUT THE IUT INTO THE NEXT LOCATION.
013776  030502                    BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
014000  001362                    BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
014002  004767  001500            JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
```

```
                                ;*****************************************************************
                                ;*TEST 31        "BRANCH GOBBLE" TEST
                                ;* THIS TEST LOADS THE ROUTINE FOUND AT "BRGOB" BELOW INTO THE BEGINNING
                                ;*  OF MEMORY UNDER TEST.  IF THERE IS ROOM, IT LOADS THE ROUTINE AGAIN
                                ;*  STARTING AT THE LAST INSTRUCTION OF THE FIRST COPY, THUS OVERLAYING
                                ;*  THE EXIT INSTRUCTION (JMP (R1)) OF THE FIRST COPY.  THIS CONTINUES
                                ;*  UNTIL THE CURRENT BANK(S) ARE FULL.  THE CODE IS THEN EXECUTED.
                                ;*  THIS ENTIRE PROCESS IS THEN REPEATED STARTING AT THE SECOND ADDRESS
                                ;*  OF THE BANK UNDER TEST, THEN THE THIRD, ETC., UNTIL EACH INSTRUCTION
                                ;*  OF THE ROUTINE HAS BEEN LOADED INTO AND EXECUTED OUT OF EVERY LOCATION.
                                ;*****************************************************************
014006                          TST31:
014006  004567  005034                      JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
014012  000077                              .WORD   77              ;MINIMUM BLOCK SIZE OF 32. WORDS
                                                                    ;  REQUIRED FOR THIS TEST.
014014  000167  000326                      JMP     TST32           ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                                                                    ;  AVAILABLE FOR TEST.
014020  012767  000031  165160              MOV     #$TN-1,$TESTN   ;;SET TEST NUMBER IN  MAIL BOX
014026                          BRGOBL:
014026  004467  000676                      JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
014032  012746  000024          BGTOP:      MOV     #BGLEN-1, -(SP) ;SAVE ROUTINE LENGTH
014036  010246                  BGLOOP:     MOV     R2,     -(SP)   ;SAVE 1ST BLOCK ADDRESS.
014040  016700  165554          5$:         MOV     .BRGOB, R0      ;GET FIRST LOC OF "BRANCH GOBBLE" ROUTINE.
014044  012701  000025                      MOV     #BGLEN, R1      ;SET UP LENGTH COUNTER
014050  012022                  6$:         MOV     (R0)+,  (R2)+   ;MOV ROUTINE INTO MEMORY
014052  030502                              BIT     R5,     R2      ;CHECK FOR END OF BLOCK.
014054  001404                              BEQ     7$              ;BRANCH WHEN DONE.
014056  005301                              DEC     R1              ;COUNT WORDS IN ROUTINE.
014060  001373                              BNE     5$              ;BR IF MORE.
014062  005742                              TST     -(R2)           ;BACK UP ONE TO OVERLAY RETURN.
014064  000765                              BR      5$              ;BR TO START ANOTHER ROUTINE.
014066  005301                  7$:         DEC     R1              ;CHECK FOR EXACT FINISH.
014070  001407                              BEQ     9$              ;BR IF ALL OK.
014072  162701  000025                      SUB     #BGLEN, R1      ;SET COUNTER FOR BACKING UP.
014076  005042                  8$:         CLR     -(R2)           ;PUT HALTS IN EXTRA MEMORY.
014100  005201                              INC     R1              ;COUNT NUMBER OF WORDS TO BACK UP.,
014102  001375                              BNE     8$              ;BR IF MORE.
014104  016722  000064                      MOV     BGEND,  (R2)+   ;INSERT RETURN INSTRUCTION.
014110  011602                  9$:         MOV     (SP),   R2      ;RESET POINTER TO BEGINNING OF BLOCK.
014112  016700  165504                      MOV     .BGERR, R0      ;POINT TO ERROR ROUTINE.
014116  016701  165502                      MOV     .BGEXI, R1      ;POINT TO EXIT ROUTINE.
014122  000112                              JMP     (R2)            ;GO TO BRANCH GOBBLE ROUTINES IN MEMORY.

                                ;*  "BRANCH GOBBLE" ROUTINE WHICH IS RELOCATED INTO MEMORY UNDER TEST
                                ;*    AND THEN EXDIEICUTED THERE.
014124  010703                  BRGOB:      MOV     PC,     R3      ;GET CURRENT ROUTINE LOCATION.
014126  062703  000024          BG1:        ADD     #BG3-BG1, R3    ;POINT TO DATA WORD "BG3"
014132  010304                              MOV     R3,     R4      ;COPY
014134  005204                              INC     R4              ;POINT TO DATA BYTE "BG4"
014136  005013                              CLR     (R3)            ;CLR DATA WORD "BG3" & "BG4"
014140  000261                  BG2:        SEC                     ;SET CARRY...TO BE ADDED TO "BG3"
014142  105513                              ADCB    (R3)            ;ADD THE CARRY... AS IF INC INST
014144  100403                              BMI     BG5             ;BRANCH WHEN BIT 7 IS SET
014146  105214                              INCB    (R4)
014150  000773                              BR      BG2             ;BR BACK AND CONTINUE TO COUNT.
014152     000                  BG3:        .BYTE   0               ;DATA BYTE POINTED TO BY R3
```

```
3684  014153  000                    BG4:    .BYTE   0               ;DATA BYTE POINTED TO BY R4
3685  014154  102401                 BG5:    BVS     BG6             ;BRANCH IF V-BIT SET...IT SHOULD BE.
3686  014156  004510                         JSR     R5,     (R0)    ;ERROR!!!       V-BIT NOT SET.
3687  014160  000242                 BG6:    CLV                     ;CLR V-BIT
3688  014162  105214                         INCB    (R4)            ;ONE MORE INC ON "BG4"
3689  014164  103402                         BCS     BG7             ;BR IF C SET...IT SHOULD NOT BE.
3690  014166  102001                         BVC     BG7             ;BR IF V SET...IT SHOULD NOT BE.
3691  014170  100401                         BMI     BGEND           ;BR IF N SET...IT SHOULD BE.
3692  014172  004510                 BG7:    JSR     R5,     (R0)    ;ERROR!!!       CC NOT = 2
3693  014174  000111                 BGEND:  JMP     (R1)            ;EXIT CODE...ONLY AT END OF BLOCK.
3694           000025                 BGLEN=  <.-BRGOB>/2             ;CALCULATED LENGTH OF ROUTINE IN WORDS.
3695
3696  014176  013767  177776  164760 BGERR:  MOV     @#PSW,  $TMP3   ;SAVE PSW
3697  014204  010567  164746         MOV     R5,     $TMP0           ;SAVE ERROR PC
3698  014210  160367  164742         SUB     R3,     $TMP0           ;OFFSET IT TO THE DATA POINTER
3699  014214  062767  014150  164734 ADD     #BG3-2, $TMP0           ;MAKE IT VIRTUAL SO IT CAN BE FOUND IN LISTING
3700  014222  010567  164732         MOV     R5,     $TMP1           ;GET ERROR PC
3701  014226  162767  000002  164724 SUB     #2,     $TMP1           ;MAKE IT PHYSICAL
3702  014234  012767  000012  164720 MOV     #12,    $TMP2           ;SET UP S/B PSW
3703  014242  010367  164652         MOV     R3,     $GDADR          ;GET PHYSICAL DATA LOCATION
3704  014246  011367  164654         MOV     (R3),   $BDDAT          ;GET THE DATA
3705  014252  026727  164700  014172 CMP     $TMP0,  #BG7            ;CHECK WHICH ERROR
3706  014260  001404                 BEQ     1$                      ;BRANCH IF SECOND ERROR
3707  014262  012767  077600  164634 MOV     #077600,$GDDAT          ;SET UP S/B DATA
3708  014270  000403                 BR      2$                      ;SKIP
3709  014272  012767  100200  164624 1$:     MOV     #100200,$GDDAT  ;SET UP S/B DATA
3710  014300                         2$:
3711  014300  004767  005532         JSR     PC,     $ERROR          ;*** ERROR *** (GO TYPE A MESSAGE)
3712  014304  000022                 .WORD   22                      ;ERROR TYPE CODE.
3713  014306  000205                 RTS     R5                      ;CONTINUE TESTING.
3714
3715                                 ;*  BRANCH GOBBLE EXIT ROUTINE
3716  014310  012602                 BGEXIT: MOV     (SP)+,  R2      ;GET FIRST ADDRESS OF BLOCK (SHIFTED).
3717  014312  005722                         TST     (R2)+           ;SHIFT IT ONE WORD.
3718  014314  005767  164670                 TST     $PASS           ;CHECK FOR PASS 0.
3719  014320  001402                         BEQ     1$              ;BR IF FIRST PASS
3720  014322  005316                         DEC     (SP)            ;COUNT TIMES SHIFTED
3721  014324  001244                         BNE     BGLOOP          ;BR IF NOT DONE ENOUGH SHIFTS.
3722  014326  005726                 1$:     TST     (SP)+           ;ADJUST STACK POINTER.
3723  014330  050502                         BIS     R5,     R2      ;RESET ADR POINTER TO TOP OF BLOCK.
3724  014332  005202                         INC     R2              ;GO TO NEXT BLOCK
3725  014334  004767  001146                 JSR     PC,     MMUP    ;GO FIND NEXT BLOCK AND LOOP TO BGLOOP.
```

```
3726                                        .SBTTL DONE:    RELOCATE PROGRAM AND REPEAT ALL TESTS.
3727   014340                               DONE:
3728   014340  004567  004502                       JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
3729   014344  000000                               .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
3730   014346  005067  164614               TST32:  CLR     $TIMES          ;RESET ITERATION COUNTER FOR RESTARTING TEST.
3731   014352  105067  164524                       CLRB    $TSTNM          ;RESET TEST NUMBER.
3732   014356  036767  164220  165146       1$:     BIT     PRGMAP, SAVTST  ;CHECK IF PROGRAM IS IN TEST AREA.
3733   014364  001004                               BNE     2$              ;BR IF IT PROG IN MEM TO BE TESTED.
3734   014366  036767  164212  165140               BIT     PRGMAP+2,SAVTST+2 ;CHECK HI 64K
3735   014374  001426                               BEQ     $EOP            ;BR IF PROG NOT IN MEM TO BE TESTED.
3736   014376  032777  000200  164532       2$:     BIT     #SW07,  @SWR    ;CHECK FOR INHIBIT RELOCATION SWITCH.
3737   014404  001022                               BNE     $EOP            ;SKIP RELOCATION IF SWITCH SET.
3738   014406  022767  000003  164166               CMP     #3,     PRGMAP  ;CHECK IF PROGRAM IN FIRST 8K.
3739   014414  001007                               BNE     4$              ;BR IF NOT IN FIRST 8K.
3740   014416  005737  000042                       TST     @#42            ;CHECK FOR A MONITOR.
3741   014422  001006                               BNE     5$              ;BR IF A MONITOR.
3742   014424  004767  002332                       JSR     PC,     RELTOP  ;RELOCATE PROGRAM TO TOP OF MEMORY.
3743   014430  000167  170730               3$:     JMP     START1          ;LOOP BACK AND RUN ALL TESTS AGAIN.
3744
3745   014434  004767  002724               4$:     JSR     PC,     RELO    ;RELOCATE PROGRAM BACK TO FIRST 8K.
3746   014440  004767  003126               5$:     JSR     PC,     RESLDR  ;RESTORE LOADERS.
3747   014444  004567  006570                       JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
3748   014450  001177                               .WORD   $CRLF           ;ADDRESS OF MESSAGE TO BE TYPED
```

```
3749                                    ;*****************************************************************
3750
3751                                    .SBTTL  END OF PASS ROUTINE
3752
3753                                    ;*INCREMENT THE PASS NUMBER ($PASS)
3754                                    ;*TYPE "END PASS #XXXXX" (WHERE XXXXX IS A DECIMAL NUMBER)
3755                                    ;*IF THERES A MONITOR GO TO IT
3756                                    ;*IF THERE ISN'T JUMP TO START1
3757
3758  014452                            $EOP:
3759  014452  000240                           NOP
3760  014454  005067  164506                   CLR     $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
3761  014460  005267  164524                   INC     $PASS           ;;INCREMENT THE PASS NUMBER
3762  014464  042767  100000  164516          BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
3763  014472  005327                          DEC     (PC)+           ;;LOOP?
3764  014474  000001                  $EOPCT: .WORD   1
3765  014476  003027                          BGT     $DOAGN          ;;YES
3766  014500  012737                          MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
3767  014502  000001                  $ENDCT: .WORD   1
3768  014504  014474                          $EOPCT
3769  014506  004567  006526                   JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
3770  014512  014562                           .WORD   $ENDMG          ;ADDRESS OF MESSAGE TO BE TYPED
3771  014514  016746  164470                   MOV     $PASS,-(SP)     ;;SAVE $PASS FOR TYPEOUT
3772                                    ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPDS ROUTINE
3773                                    ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
3774  014520  013746  177776                   MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
3775  014524  004767  007424                   JSR     PC,     $TYPDS  ;GO TO THE SUBROUTINE
3776  014530  004567  006504                   JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
3777  014534  014577                           .WORD   $ENULL          ;ADDRESS OF MESSAGE TO BE TYPED
3778  014536                            $GET42:
3779
3780  014536  016700  163300                   MOV     42,     R0              ;;GET MONITOR ADDRESS
3781  014542  001405                          BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
3782  014544  000005                          RESET                   ;;CLEAR THE WORLD
3783  014546  004710                  $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
3784  014550  000240                          NOP                     ;;SAVE ROOM
3785  014552  000240                          NOP                     ;;FOR
3786  014554  000240                          NOP                     ;;ACT11
3787  014556                            $DOAGN:
3788  014556  000167  170602                   JMP     START1          ;;RETURN*****
3789  014562  005015  047105  020104  $ENDMG: .ASCIZ   <15><12>/END PASS #/
3790  014570  040520  051523  021440
3791  014576     000
3792  014577     377     377     000  $ENULL: .BYTE   -1,-1,0          ;;NULL CHARACTER STRING
```

```
                                        .SBTTL SUBROUTINE AND TRAP ROUTINE SECTION.
                                        .SBTTL  MEMORY MANAGEMENT AND ADDRESSING SUBROUTINES.
                                        ;*****************************************************************
                                        ;*  SET UP ALL THE MEM MGMT REGISTERS FOR NORMAL OPERATION.
                                        ;*  THE PROGRAM IS POINTED TO BY PARS 0 AND 1.
                                        ;*  THE MEMORY UNDER TEST IS POINTED TO BY PARS 2 AND 3.
                                        ;*  THE DEVICE ADDRESS AREA IS POINTED TO BY PAR 7.
                                        ;*  PARS 4, 5, AND 6 ARE UNUSED.
                                        ;*****************************************************************
014602                                  MMINIT:
014602  012737  077406  172300                 MOV     #200-1*400+UP+RW,@#KIPDR0       ;SET KIPDR0 = RW UP 200 BLOCKS
014610  012737  077406  172302                 MOV     #200-1*400+UP+RW,@#KIPDR1       ;SET KIPDR1 = RW UP 200 BLOCKS
014616  012737  077406  172304                 MOV     #200-1*400+UP+RW,@#KIPDR2       ;SET KIPDR2 = RW UP 200 BLOCKS
014624  012737  077406  172306                 MOV     #200-1*400+UP+RW,@#KIPDR3       ;SET KIPDR3 = RW UP 200 BLOCKS
014632  005037  172310                        CLR     @#KIPDR4
014636  005037  172312                        CLR     @#KIPDR5
014642  005037  172314                        CLR     @#KIPDR6
014646  012737  077406  172316                 MOV     #200-1*400+UP+RW,@#KIPDR7       ;SET KIPDR7 = RW UP 200 BLOCKS
014654  005037  172340                        CLR     @#KIPAR0        ;MAP PAR0 INTO BANK0
014660  012737  000200  172342                 MOV     #200, @#KIPAR1  ;MAP PAR1 INTO BANK1
014666  005037  172344                        CLR     @#KIPAR2        ;MAP PAR2 INTO BANK0
014672  005037  172346                        CLR     @#KIPAR3
014676  005037  172350                        CLR     @#KIPAR4
014702  005037  172352                        CLR     @#KIPAR5
014706  005037  172354                        CLR     @#KIPAR6
014712  012737  007600  172356                 MOV     #7600, @#KIPAR7 ;MAP PAR7 INTO I/O BANK
014720  012737  000001  177572                 MOV     #1,     @#SR0   ;ENABLE MEMORY MANAGEMENT
014726  000207                                RTS     PC              ;RETURN


                                        ;*****************************************************************
                                        ;*  MEMORY ADDRESS POINTER INITIALIZATION ROUTINES.
                                        ;*****************************************************************
014730  012767  000001  164604  INITMM: MOV     #BIT0,  BITPT   ;SET POINTER TO BANK0
014736  005067  164602                 CLR     BITPT+2         ;CLEAR HI 64K BANK POINTERS
014742  005002                        CLR     R2              ;SET ADDRESS POINTER TO 0
014744  016705  164634                 MOV     BLKMSK, R5      ;RESET R5 TO BLOCK MASK.
014750  005767  163632                 TST     MMAVA           ;CHECK FOR MEM MGMT AVAILABLE
014754  001514                        BEQ     10$             ;BRANCH IF NO MEM MGMT
014756  005037  172344                 CLR     @#KIPAR2        ;SET UP 3RD PAR TO BANK0
014762  012702  040000                 MOV     #40000, R2      ;RESET VIRTUAL ADR POINTER
014766  036767  164550  164532  1$:    BIT     BITPT,  TSTMAP  ;CHECK IF THIS BANK TO BE TESTED
014774  001015                        BNE     2$              ;BRANCH IF MATCH
014776  036767  164542  164524         BIT     BITPT+2,TSTMAP+2 ;CHECK IN HI MAP
015004  001011                        BNE     2$              ;BRANCH IF MATCH
015006  062737  000200  172344         ADD     #200,   @#KIPAR2 ;UPDATE MEM MGMT, THIRD PAR.
015014  006367  164522                 ASL     BITPT           ;UPDATE LO POINTER TO NEXT BANK.
015020  006167  164520                 ROL     BITPT+2         ;...HI POINTER.
015024  100360                        BPL     1$              ;BR IF MORE.
015026  000000                        HALT                    ;FATAL ERROR!!!  NO 4K BANK FOUND?
015030  036767  164506  164542  2$:    BIT     BITPT,  LADMAP  ;CHECK IF LAST BANK.
015036  001004                        BNE     3$              ;BR IF LAST BANK.
015040  036767  164500  164534         BIT     BITPT+2,LADMAP+2 ;CHECK IF LAST BANK.
015046  001405                        BEQ     4$              ;BR IF NOT LAST BANK.
015050  016705  164522          3$:    MOV     LADMSK, R5      ;SET MASK TO FIND LAST ADR.
015054  042767  020000  164512         BIC     #20000, TMPLAD  ;MAKE SURE VIRTUAL LAST ADR IN BANK 2.
```

```
3849  015062  013737  172344  172346  4$:     MOV     @#KIPAR2,@#KIPAR3 ;COPY CURRENT PAR INTO FORTH PAR.
3850  015070  016767  164446  164450          MOV     BITPT,  TMPPT     ;COPY BITPT...LO 64K.
3851  015076  016767  164442  164444          MOV     BITPT+2,TMPPT+2   ;...HI 64K.
3852  015104  032705  020000                  BIT     #BIT13, R5        ;CHECK FOR A BLOCK SIZE OF 8K.
3853  015110  001505                          BEQ     21$               ;BRANCH IF NOT 8K.
3854  015112  062737  000200  172346  5$:     ADD     #200,   @#KIPAR3  ;UP DATE FORTH PAR.
3855  015120  006367  164422                  ASL     TMPPT             ;UPDATE LO POINTER TO NEXT 4K BANK.
3856  015124  006167  164420                  ROL     TMPPT+2           ;...HI POINTER.
3857  015130  100473                          BMI     20$               ;BR IF NO MORE.
3858  015132  036767  164410  164366          BIT     TMPPT,  TSTMAP    ;CHECK IF BANK TO BE TESTED.
3859  015140  001004                          BNE     6$                ;BRANCH IF A MATCH.
3860  015142  036767  164402  164360          BIT     TMPPT+2,TSTMAP+2  ;CHECK FOR HI 64K BANKS.
3861  015150  001760                          BEQ     5$                ;BRANCH IF NO MEMORY
3862  015152  036767  164370  164420  6$:     BIT     TMPPT,  LADMAP    ;CHECK IF LAST BANK.
3863  015160  001004                          BNE     7$                ;BRANCH IF A MATCH
3864  015162  036767  164362  164412          BIT     TMPPT+2,LADMAP+2  ;CHECK HI 64K
3865  015170  001455                          BEQ     21$               ;BR IF NOT LAST BANK.
3866  015172  016705  164400          7$:     MOV     LADMSK, R5        ;RESET MASK TO FIND LAST ADR.
3867  015176  052767  020000  164370          BIS     #20000, TMPLAD    ;MAKE SURE LAST ADDRESS IS IN BANK 3.
3868  015204  000447                          BR      21$               ;BR TO FINISH UP.
3869
3870  015206  036767  164330  164312  10$:    BIT     BITPT,  TSTMAP    ;CHECK IF THIS BANK TO BE TESTED.
3871  015214  001006                          BNE     11$               ;BR IF MATCH.
3872  015216  062702  020000                  ADD     #20000, R2        ;UPDATE PHYSICAL ADR PNTR TO NEXT BANK.
3873  015222  106367  164314                  ASLB    BITPT             ;UPDATE BANK POINTER TO NEXT BANK.
3874  015226  100367                          BPL     10$               ;BR IF MORE BANKS.
3875  015230  000000                          HALT                      ;FATAL ERROR!!!  NO 4K BANK FOUND?
3876  015232  016767  164304  164306  11$:    MOV     BITPT, TMPPT      ;COPY BANK POINTER.
3877  015240  036767  164276  164332          BIT     BITPT,  LADMAP    ;CHECK IF LAST BANK.
3878  015246  001021                          BNE     12$               ;BR IF LAST BANK.
3879  015250  032705  020000                  BIT     #BIT13, R5        ;CHECK FOR 8K BLOCK SIZE.
3880  015254  001423                          BEQ     21$               ;BRANCH IF SMALLER BLOCK SIZE.
3881  015256  106367  164264                  ASLB    TMPPT             ;POINT TO NEXT BANK.
3882  015262  100416                          BMI     20$               ;BRANCH IF OVERFLOW.
3883  015264  036767  164256  164234          BIT     TMPPT,  TSTMAP    ;CHECK IF BANK TO BE TESTED.
3884  015272  001412                          BEQ     20$               ;BRANCH IF NOT TO BE TESTED.
3885  015274  112767  000011  164253          MOVB    #11,    FLAG8K    ;SET 8K BLOCK SIZE FLAG.
3886  015302  036767  164240  164270          BIT     TMPPT,  LADMAP    ;CHECK FOR LAST BANK.
3887  015310  001403                          BEQ     20$               ;BR IF NOT LAST BANK.
3888  015312  016705  164260          12$:    MOV     LADMSK, R5        ;RESET MASK TO FIND LAST ADR.
3889  015316  000402                          BR      21$               ;SKIP MASK RESET.
3890  015320  012705  017777          20$:    MOV     #MASK4K,R5        ;RESET MASK TO 4K BLOCK SIZE.
3891  015324  056767  164212  164214  21$:    BIS     BITPT,  TMPPT     ;SET TMPPT FOR FLAGING LAST BANK.
3892  015332  056767  164206  164210          BIS     BITPT+2,TMPPT+2
3893  015340  036767  164176  164220          BIT     BITPT,  FADMAP    ;CHECK IF FIRST ADDRESS NEEDS TO BE SET.
3894  015346  001004                          BNE     22$               ;BR IF FIRST BANK.
3895  015350  036767  164170  164212          BIT     BITPT+2,FADMAP+2  ;CHECK HI 64K.
3896  015356  001450                          BEQ     INITEX            ;BR IF NOT FIRST BANK.
3897  015360  016702  164176          22$:    MOV     TMPFAD, R2        ;RESET ADDRESS POINTER TO FIRST ADR.
3898  015364  000445                          BR      INITEX
3899
3900  015366  016705  164212          INITDN: MOV     BLKMSK, R5        ;RESET R5 TO CURRENT BLOCK MASK.
3901  015372  005002                          CLR     R2                ;INIT ADDRESS POINTER.
3902  015374  005767  163206                  TST     MMAVA             ;CHECK FOR MEM MGMT
3903  015400  001411                          BEQ     31$               ;BRANCH IF NO MEM MGMT
3904  015402  012767  100000  164134          MOV     #BIT15, BITPT+2   ;SET POINTER TO TOP BIT
```

```
3905  015410  005067  164126                    CLR     BITPT
3906  015414  012737  007600  172344            MOV     #7600,  @#KIPAR2 ;SET PAR TO TOP OF MEM
3907  015422  000403                            BR      32$              ;BRANCH TO COMMON AREA
3908
3909  015424  012767  000400  164110  31$:      MOV     #BIT8,  BITPT    ;SET UP BANK POINTER
3910  015432  012767  015454  164112  32$:      MOV     #33$,   MMORE    ;SET "MMDOWN" EXIT ADDRESS.
3911  015440  066767  163134  164104            ADD     RELOCF, MMORE    ;ADD OFFSET
3912  015446  004767  000524                    JSR     PC,     MMDOWN   ;ROUTINE TO SEARCH DOWNWARD FOR TOP MEM BANK
3913  015452  000000                            HALT                     ;FATAL ERROR!!!  NO MEM INDICATED IN MEM MAP ABOVE 8K!
3914  015454  036767  164062  164116  33$:      BIT     BITPT,  LADMAP   ;CHECK FOR NON BOUNDRY LAST ADDR.
3915  015462  001004                            BNE     34$              ;BR IF LAST BANK FLAG FOUND.
3916  015464  036767  164054  164110            BIT     BITPT+2,LADMAP+2 ;CHECK FOR NON BOUNDRY LAST ADDR.
3917  015472  001402                            BEQ     INITEX           ;BR IF NO LAD FLG FOUND.
3918  015474  016702  164072          34$:      MOV     LSTADR, R2       ;SET UP R2.
3919  015500  010467  164046          INITEX:   MOV     R4,     MMORE    ;PUT RETURN PC INTO "MMORE"
3920  015504  000204                            RTS     R4               ;RETURN
3921
3922                                  ;*****************************************************************
3923                                  ;*  COMMON UPWARDS ADDRESSING ROUTINE
3924                                  ;*  FINDS NEXT EXISTING 4K BANK AND UPDATES POINTERS.
3925                                  ;*  GOES TO ADDRESS IN "MMORE" IF MORE BANKS,
3926                                  ;*  DOES STRAIGHT EXIT WHEN ALL MEMORY HAS BEEN DONE.
3927                                  ;*****************************************************************
3928  015506  036767  164034  164064  MMUP:     BIT     TMPPT,  LADMAP   ;CHECK FOR LAST BANK FLAG.
3929  015514  001122                            BNE     10$              ;BR IF LAST BANK.
3930  015516  036767  164026  164056            BIT     TMPPT+2,LADMAP+2 ;CHECK FOR LAST BANK FLAG.
3931  015524  001116                            BNE     10$              ;BR IF LAST BANK.
3932  015526  016705  164052                    MOV     BLKMSK, R5       ;RESET R5 TO BLOCK MASK.
3933  015532  005767  163050                    TST     MMAVA            ;CHECK FOR MEM MGMT AVAILABLE
3934  015536  001515                            BEQ     20$              ;BRANCH IF NO MEM MGMT
3935  015540  012702  040000                    MOV     #40000, R2       ;RESET VIRTUAL ADR POINTER
3936  015544  062737  000200  172344  1$:       ADD     #200,   @#KIPAR2 ;UPDATE MEM MGMT, THIRD PAR.
3937  015552  006367  163764                    ASL     BITPT            ;UPDATE LO POINTER TO NEXT BANK.
3938  015556  006167  163762                    ROL     BITPT+2          ;...HI POINTER.
3939  015562  100577                            BMI     32$              ;BR IF ALL DONE.
3940  015564  036767  163752  163734            BIT     BITPT,  TSTMAP   ;CHECK IF THIS BANK EXISTS
3941  015572  001004                            BNE     2$               ;BRANCH IF MATCH
3942  015574  036767  163744  163726            BIT     BITPT+2,TSTMAP+2 ;CHECK IN HI MAP
3943  015602  001760                            BEQ     1$               ;BRANCH IF NO MATCH
3944  015604  036767  163732  163766  2$:       BIT     BITPT,  LADMAP   ;CHECK FOR LAST BANK FLAG.
3945  015612  001004                            BNE     3$               ;BRANCH IF LAST BANK FLAG.
3946  015614  036767  163724  163760            BIT     BITPT+2,LADMAP+2 ;CHECK IF LAST BANK FLAG.
3947  015622  001405                            BEQ     4$               ;BR IF NOT LAST BANK.
3948  015624  016705  163746          3$:       MOV     LADMSK, R5       ;RESET MASK.
3949  015630  042767  020000  163736            BIC     #20000, TMPLAD   ;MAKE SURE VIRTUAL LAST ADR IN BANK 2
3950  015636  016767  163700  163702  4$:       MOV     BITPT,  TMPPT    ;COPY BITPT...LO 64K.
3951  015644  016767  163674  163676            MOV     BITPT+2,TMPPT+2  ;...HI 64K.
3952  015652  032705  020000                    BIT     #BIT13, R5       ;CHECK FOR A BLOCK SIZE OF 8K.
3953  015656  001530                            BEQ     31$              ;BRANCH IF NOT.
3954  015660  013737  172344  172346            MOV     @#KIPAR2,@#KIPAR3 ;COPY CURRENT PAR INTO FORTH PAR.
3955  015666  062737  000200  172346  5$:       ADD     #200,   @#KIPAR3 ;UP DATE FORTH PAR.
3956  015674  006367  163646                    ASL     TMPPT            ;UPDATE LO POINTER TO NEXT 4K BANK.
3957  015700  006167  163644                    ROL     TMPPT+2          ;...HI POINTER.
3958  015704  100513                            BMI     30$              ;BR IF NO MORE.
3959  015706  036767  163634  163612  6$:       BIT     TMPPT,  TSTMAP   ;CHECK IF BANK TO BE TESTED.
3960  015714  001004                            BNE     7$               ;BRANCH IF A MATCH.
```

```
 3961  015716  036767  163626  163604          BIT     TMPPT+2,TSTMAP+2 ;CHECK FOR HI 64K BANKS.
 3962  015724  001760                          BEQ     5$               ;BRANCH IF NO MEMORY
 3963  015726  036767  163614  163644  7$:     BIT     TMPPT,LADMAP     ;CHECK FOR LAST BANK FLAG.
 3964  015734  001004                          BNE     8$               ;BRANCH IF A MATCH
 3965  015736  036767  163606  163636          BIT     TMPPT+2,LADMAP+2 ;CHECK HI 64K
 3966  015744  001475                          BEQ     31$              ;BR IF NO LAST BANK FLAG.
 3967  015746  016705  163624          8$:     MOV     LADMSK, R5       ;RESET MASK TO FIND LAST ADDRESS.
 3968  015752  052767  020000  163614          BIS     #20000, TMPLAD   ;SET VIRTUAL ADR TO BANK 3.
 3969  015760  000467                          BR      31$
 3970
 3971  015762  026702  163606          10$:    CMP     TMPLAD, R2       ;CHECK IF LAST ADR REACHED.
 3972  015766  001064                          BNE     31$              ;BR IF MORE.
 3973  015770  000474                          BR      32$              ;BR IF ALL DONE.
 3974
 3975  015772  106267  163557          20$:    ASRB    FLAG8K           ;SHIFT 8K FLAG
 3976  015776  001407                          BEQ     22$              ;BR IF NOT 8K BLOCK.
 3977  016000  103455                          BCS     30$              ;BR IF ANOTHER 4K.
 3978  016002  105067  163547                  CLRB    FLAG8K           ;CLEAR OUT ALL FLAGS.
 3979  016006  162702  040000                  SUB     #40000, R2       ;BACK UP 8K.
 3980  016012  062702  020000          21$:    ADD     #20000, R2       ;UPDATE PHYSICAL ADR PNTR TO NEXT BANK.
 3981  016016  106367  163520          22$:    ASLB    BITPT            ;UPDATE POINTER.
 3982  016022  100457                          BMI     32$              ;BRANCH WHEN END IS REACHED.
 3983  016024  036767  163512  163474          BIT     BITPT,  TSTMAP   ;CHECK IF THIS BANK EXISTS.
 3984  016032  001767                          BEQ     21$              ;BRANCH IF NO MATCH.
 3985  016034  036767  163502  163536          BIT     BITPT,  LADMAP   ;CHECK FOR LAST BANK FLAG.
 3986  016042  001402                          BEQ     23$              ;BR IF NO MATCH.
 3987  016044  016705  163526                  MOV     LADMSK, R5       ;RESET MASK TO FIND LAST ADR.
 3988  016050  016767  163466  163470  23$:    MOV     BITPT,  TMPPT    ;SET UP TMP POINTER.
 3989  016056  032705  020000                  BIT     #BIT13, R5       ;CHECK FOR 8K BLOCK SIZE.
 3990  016062  001426                          BEQ     31$              ;BRANCH IF SMALLER BLOCK SIZE.
 3991  016064  106367  163456                  ASLB    TMPPT            ;POINT TO NEXT BANK.
 3992  016070  100421                          BMI     30$              ;BRANCH IF OVERFLOW.
 3993  016072  036767  163450  163426          BIT     TMPPT,  TSTMAP   ;CHECK IF BANK TO BE TESTED.
 3994  016100  001415                          BEQ     30$              ;BRANCH IF NOT TO BE TESTED.
 3995  016102  036767  163434  163470          BIT     BITPT,  LADMAP   ;CHECK FOR LAST BANK FLAG.
 3996  016110  112767  000011  163437          MOVB    #11,    FLAG8K   ;SET 8K BLOCK FLAG.
 3997  016116  036767  163420  163454          BIT     BITPT,  LADMAP   ;CHECK FOR LAST BANK FLAG.
 3998  016124  001403                          BEQ     30$              ;BR IF NO FLAG.
 3999  016126  016705  163444                  MOV     LADMSK, R5       ;RESET MASK TOFIND LAST ADR.
 4000  016132  000402                          BR      31$
 4001  016134  012705  017777          30$:    MOV     #MASK4K,R5       ;SET MASK TO 4K.
 4002  016140  056767  163376  163400  31$:    BIS     BITPT,  TMPPT    ;SET TMPPT FOR FINDING LAST ADR.
 4003  016146  056767  163372  163374          BIS     BITPT+2,TMPPT+2
 4004  016154  016716  163372                  MOV     MMORE,  (SP)     ;FUDGE RETURN ADDRESS TO LOOP.
 4005  016160  000207                          RTS     PC               ;RETURN
 4006                          ;*  BEFORE FINAL EXIT, CHECK FOR ANY NON-TRAP PARITY ERRORS.
 4007  016162  005767  164064          32$:    TST     MPRX             ;CHECK FOR ANY PARITY REGISTERS PRESENT.
 4008  016166  001402                          BEQ     33$              ;BR IF NONE.
 4009  016170  004767  001660                  JSR     PC,     CKPMER   ;CHECK FOR PARITY MEMORY ERRORS.
 4010  016174  000207                  33$:    RTS     PC               ;STRAIGHT RETURN.
 4011
 4012                          ;*****************************************************************
 4013                          ;*  MEMORY DOWNWARDS ADDRESSING SUBROUTINE.
 4014                          ;*  FINDS NEXT LOWER 4K BANK AND UPDATES POINTERS.
 4015                          ;*  GOES TO ADDRESS IN "MMORE" IF MORE BANKS,
 4016                          ;*  DOES STRAIGHT EXIT WHEN ALL MEMORY HAS BEEN DONE.
```

```
4017                                    ;*****************************************************************
4018  016176  036767  163340  163362  MMDOWN: BIT     BITPT,  FADMAP  ;CHECK FOR FIRST ADR FLAG.
4019  016204  001004                          BNE     1$              ;BR IF FIRST ADR IN THIS BANK.
4020  016206  036767  163332  163354          BIT     BITPT+2,FADMAP+2 ;CHECK FOR FIRST ADR FLAG.
4021  016214  001404                          BEQ     2$              ;BR IF NO FLAG
4022  016216  026702  163340          1$:     CMP     TMPFAD, R2      ;CHECK IF FIRST ADDRESS REACHED.
4023  016222  001052                          BNE     9$              ;BR IF MORE.
4024  016224  000453                          BR      10$             ;BR IF ALL DONE.
4025  016226  005767  162354          2$:     TST     MMAVA           ;CHECK IF MEM MGMT IS AVAILABLE
4026  016232  001425                          BEQ     6$              ;BRANCH IF NOT
4027  016234  162737  000200  172344  3$:     SUB     #200, @#KIPAR2  ;LOWER MEM MGMT PAR BY 4K
4028  016242  006067  163276                  ROR     BITPT+2         ;MOV POINTER TO NEXT LOWER BANK...HI MAP.
4029  016246  006067  163270                  ROR     BITPT           ;...LO MAP.
4030  016252  103440                          BCS     10$             ;BR IF NO MORE.
4031  016254  036767  163262  163244          BIT     BITPT,  TSTMAP  ;CHECK FOR BANK EXISTING
4032  016262  001004                          BNE     4$              ;BR IF BANK TO BE TESTED.
4033  016264  036767  163254  163236          BIT     BITPT+2,TSTMAP+2 ;CHECK FOR BANK IN HI MAP.
4034  016272  001760                          BEQ     3$              ;BR IF NOT THERE.
4035  016274  012702  060000          4$:     MOV     #60000, R2      ;SET ADR POINTER TO TOP OF BANK
4036  016300  000411                          BR      7$              ;GO TO COMMON EXIT
4037  016302  162702  020000          5$:     SUB     #20000, R2      ;BACK POINTER DOWN ONE BANK
4038  016306  006267  163230          6$:     ASR     BITPT           ;MOVE POINTER TO NEXT LOWER BANK
4039  016312  103420                          BCS     10$             ;BRANCH TO EXIT IF NO MORE  MEM
4040  016314  036767  163222  163204          BIT     BITPT,  TSTMAP  ;CHECK IF BANK EXISTS
4041  016322  001767                          BEQ     5$              ;BRANCH IF BANK DOESN'T EXIST
4042  016324  036767  163212  163234  7$:     BIT     BITPT,  FADMAP  ;CHECK IF FIRST BANK FLAG.
4043  016332  001004                          BNE     8$              ;BR IF FIRST BANK.
4044  016334  036767  163204  163226          BIT     BITPT+2,FADMAP+2 ;CHECK IF FIRST BANK FLAG.
4045  016342  001402                          BEQ     9$              ;BR IF NO FLAG FOUND.
4046  016344  016705  163214          8$:     MOV     FADMSK, R5      ;SET UP R5 TO FIND FIRST ADDRESS.
4047  016350  016716  163176          9$:     MOV     MMORE,  (SP)    ;RESET RETURN ADDRESS
4048  016354  000207                  10$:    RTS     PC              ;RETURN
```

```
                                        .SBTTL  SUBROUTINES FOR ADDRESS AND WORSE CASE NOISE TESTS.
                                        ;*************************************************************
                                        ;*  SUBROUTINE TO CALCULATE PHYSICAL ADDRESS AND PUT IT IN R0 (BOTTOM 16 BITS).
                                        ;*  BITS 16 AND 17 ARE IN $TMP0.
                                        ;*************************************************************
016356  010200                        PHYADR: MOV     R2,     R0      ;VITRUAL INTO R0
016360  005067  162572                        CLR     $TMP0           ;CLEAR TEMP SAVE OF HIGH BITS
016364  005767  162216                        TST     MMAVA           ;CHECK FOR MEM MGMT AVAILABLE
016370  001417                              BEQ     1$              ;BRANCH IF NO MEM MGMT
016372  010146                              MOV     R1,-(SP)        ;;PUSH R1 ON STACK
016374  013701  172344                      MOV     @#KIPAR2, R1    ;GET PAR TO BE ADDED TO VIRTUAL
016400  006301                              ASL     R1              ;SHIFT IT 6 TIMES
016402  006301                              ASL     R1
016404  006301                              ASL     R1
016406  006301                              ASL     R1
016410  006301                              ASL     R1
016412  006167  162540                      ROL     $TMP0           ;SAVE EXTRA BITS
016416  006301                              ASL     R1
016420  006167  162532                      ROL     $TMP0
016424  060100                              ADD     R1,     R0      ;ADD SHIFTED PAR TO VIRTUAL
016426  012601                              MOV     (SP)+,R1        ;;POP STACK INTO R1
016430  000207                          1$: RTS     PC              ;RETURN

                                        ;*************************************************************
                                        ;*  SUBROUTINE TO PUT BANK NUMBER INTO R0.
                                        ;*************************************************************
016432  005000                        BANKNO: CLR     R0              ;INIT R0
016434  010146                              MOV     R1,-(SP)        ;;PUSH R1 ON STACK
016436  010246                              MOV     R2,-(SP)        ;;PUSH R2 ON STACK
016440  016701  163076                      MOV     BITPT,  R1      ;GET BANK MAP POINTER...LO 64K.
016444  016702  163074                      MOV     BITPT+2,R2      ;...HI 64K.
016450  006202                          1$: ASR     R2              ;SHIFT POINTER...HI
016452  006001                              ROR     R1              ;...LO
016454  103403                              BCS     2$              ;BR WHEN POINTER FOUND.
016456  105200                              INCB    R0              ;COUNT BANKS.
016460  100373                              BPL     1$              ;BR IF NOT OVERFLOW.
016462  000000                              HALT            ;FATAL ERROR!!!  NO POINTER FOUND.
016464                                  2$:
016464  012602                              MOV     (SP)+,R2        ;;POP STACK INTO R2
016466  012601                              MOV     (SP)+,R1        ;;POP STACK INTO R1
016470  000207                              RTS     PC              ;RETURN

                                        ;*************************************************************
                                        ;*  SUBROUTINE TO WRITE THE CONSTANT IN R0 INTO ALL OF MEMORY.
                                        ;*************************************************************
016472                                  SETCON:
016472  004467  176232                      JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
016476  010022                          2$: MOV     R0,     (R2)+   ;MOV CONSTANT INTO MEMORY
016500  030502                              BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
016502  001375                              BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
016504  004767  176776                      JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
016510  000207                              RTS     PC              ;RETURN
```

```
 4101                                   ;*****************************************************************
 4102                                   ;* ROUTINE TO ROTATE 'C' BIT THROUGH A MEMORY LOCATION.
 4103                                   ;*****************************************************************
 4104  016512  106112                  ROTATE: ROLB    (R2)            ;(R2)=177776 OR 000001
 4105  016514  106112                          ROLB    (R2)            ;(R2)=177775 OR 000002
 4106  016516  106112                          ROLB    (R2)            ;(R2)=177773 OR 000004
 4107  016520  106112                          ROLB    (R2)            ;(R2)=177767 OR 000010
 4108  016522  106112                          ROLB    (R2)            ;(R2)=177757 OR 000020
 4109  016524  106112                          ROLB    (R2)            ;(R2)=177737 OR 000040
 4110  016526  106112                          ROLB    (R2)            ;(R2)=177677 OR 000100
 4111  016530  106112                          ROLB    (R2)            ;(R2)=177777 OR 000000
 4112  016532  106122                          ROLB    (R2)+           ;(R2)=177577 OR 000200
 4113  016534  106112                          ROLB    (R2)            ;(R2)=177377 OR 000400
 4114  016536  106112                          ROLB    (R2)            ;(R2)=176777 OR 001000
 4115  016540  106112                          ROLB    (R2)            ;(R2)=175777 OR 002000
 4116  016542  106112                          ROLB    (R2)            ;(R2)=173777 OR 004000
 4117  016544  106112                          ROLB    (R2)            ;(R2)=167777 OR 010000
 4118  016546  106112                          ROLB    (R2)            ;(R2)=157777 OR 020000
 4119  016550  106112                          ROLB    (R2)            ;(R2)=137777 OR 040000
 4120  016552  106112                          ROLB    (R2)            ;(R2)=077777 OR 100000
 4121  016554  106122                          ROLB    (R2)+           ;(R2)=177777 OR 000000
 4122  016556  000207                          RTS     PC              ;RETURN
 4123
 4124                                   ;*****************************************************************
 4125                                   ;*  SUBROUTINE TO WRITE 3 XOR 9 PATTERN INTO 256. WORD BLOCK.
 4126                                   ;*****************************************************************
 4127  016560  012704  000020          W3X9:   MOV     #16.,R4         ;EACH LOOP WRITES 256. WORDS
 4128
 4129  016564  010022                  2$:     MOV     R0,(R2)+
 4130  016566  010022                          MOV     R0,(R2)+
 4131  016570  010022                          MOV     R0,(R2)+
 4132  016572  010022                          MOV     R0,(R2)+
 4133
 4134  016574  010322                          MOV     R3,(R2)+
 4135  016576  010322                          MOV     R3,(R2)+
 4136  016600  010322                          MOV     R3,(R2)+
 4137  016602  010322                          MOV     R3,(R2)+
 4138
 4139  016604  010022                          MOV     R0,(R2)+
 4140  016606  010022                          MOV     R0,(R2)+
 4141  016610  010022                          MOV     R0,(R2)+
 4142  016612  010022                          MOV     R0,(R2)+
 4143
 4144  016614  010322                          MOV     R3,(R2)+
 4145  016616  010322                          MOV     R3,(R2)+
 4146  016620  010322                          MOV     R3,(R2)+
 4147  016622  010322                          MOV     R3,(R2)+
 4148
 4149  016624  005304                          DEC     R4
 4150  016626  001356                          BNE     2$
 4151  016630  010046                          MOV     R0,     -(SP)   ;SAVE R0
 4152  016632  010300                          MOV     R3,     R0      ;PUT R3 INTO R0
 4153  016634  012603                          MOV     (SP)+,  R3      ;PUT SAVED R0 INTO R3
 4154  016636  000207                          RTS     PC              ;RETURN
```

```
                                        .SBTTL  RELOCATION SUBROUTINES.
                                        ;*************************************************************
                                        ;* ROUTINE TO RELOCATE PROGRAM CODE
                                        ;*************************************************************
016640                                  RELOC:
016640  010246                                  MOV     R2,-(SP)        ;;PUSH R2 ON STACK
016642  010346                                  MOV     R3,-(SP)        ;;PUSH R3 ON STACK
016644  010446                                  MOV     R4,-(SP)        ;;PUSH R4 ON STACK
016646  012502                          4$:     MOV     (R5)+, R2       ;GET FIRST LOCATION.
016650  012503                                  MOV     (R5)+, R3       ;GET FIRST LOCATION OF DESTINATION.
016652  012704  020000                         MOV     #20000, R4      ;SET UP 8K COUNTER.
016656  012223                          1$:     MOV     (R2)+, (R3)+    ;MOV THE DATA.
016660  005304                                  DEC     R4              ;COUNT THE WORDS.
016662  001375                                  BNE     1$              ;BR IF MORE.
016664  012704  020000                         MOV     #20000, R4      ;RESET THE COUNTER.
016670  024243                          2$:     CMP     -(R2), -(R3)    ;CHECK THE DATA JUST MOVED.
016672  001417                                  BEQ     3$              ;BR IF DATA OK.
016674  011267  162224                         MOV     (R2), $GDDAT    ;GET SOURCE DATA.
016700  011367  162222                         MOV     (R3), $BDDAT    ;GET DESTINATION DATA.
016704  010267  162210                         MOV     R2, $GDADR      ;GET SOURCE ADDRESS.
016710  010367  162206                         MOV     R3, $BDADR      ;GET DESTINATION ADDRESS.
016714  004767  003116                         JSR     PC, $ERROR      ;*** ERROR *** (GO TYPE A MESSAGE)
016720  000023                                  .WORD   23              ;ERROR TYPE CODE.
016722  000000                                  HALT                    ;FATAL ERROR!!!  RELOCATION FAILED.
016724  162705  000004                         SUB     #4, R5          ;ADJUST RETURN POINTER.
016730  000746                                  BR      4$              ;GO BACK AND TRY AGAIN.
016732  005304                          3$:     DEC     R4              ;COUNT WORDS.
016734  001355                                  BNE     2$              ;BR IF MORE.
016736  004567  004276                         JSR     R5, $PRINT      ;GO PRINT OUT THE FOLLOWING MESSAGE.
016742  026121                                  .WORD   PRELOC          ;ADDRESS OF MESSAGE TO BE TYPED
                                                                        ;"PROGRAM RELOCATED TO "
016744  010346                                  MOV     R3,     -(SP)   ;PUT THE DATA ON THE STACK.
016746  004767  005722                         JSR     PC,     $TYPAD  ;DETERMINE THE PHYSICAL ADDRESS AND TYPE IT.
016752  012604                                  MOV     (SP)+,R4        ;;POP STACK INTO R4
016754  012603                                  MOV     (SP)+,R3        ;;POP STACK INTO R3
016756  012602                                  MOV     (SP)+,R2        ;;POP STACK INTO R2
016760  000205                                  RTS     R5              ;RETURN
                                        ;*************************************************************
                                        ;*  SUBROUTINE TO MOVE PROGRAM FROM BOTTOM TO TOP OF MEMORY.
                                        ;*************************************************************
016762  022767  000003  161612         RELTOP: CMP     #3,     PRGMAP  ;CHECK THAT THE PROGRAM IS NOW IN BANKS 0 AND 1.
016770  001401                                  BEQ     1$              ;BR IF OK
016772  000000                                  HALT                    ;FATAL ERROR!!!  PROG SHOULD BE IN BANKS 0 AND 1
016774                                  1$:
016774  010046                                  MOV     R0,-(SP)        ;;PUSH R0 ON STACK
016776  010146                                  MOV     R1,-(SP)        ;;PUSH R1 ON STACK
017000  005767  161602                         TST     MMAVA
017004  001465                                  BEQ     10$
017006  012737  007600  172346                 MOV     #7600,  @#KIPAR3 ;SET PAR TO TOP OF MEM
017014  005000                                  CLR     R0              ;INIT BANK POINTER...LO 64K
017016  012701  100000                         MOV     #BIT15, R1      ;...HI 64K.
017022  162737  000200  172346         2$:     SUB     #200,   @#KIPAR3        ;BACK DOWN ONE BANK.
017030  006001                                  ROR     R1              ;MOVE POINTER...HI 64K.
017032  006000                                  ROR     R0              ;..LO 64K.
017034  103500                                  BCS     90$
017036  030167  162462                         BIT     R1,     MEMMAP+2 ;CHECK FOR BANK EXISTS.
```

```
4211  017042  001003                        BNE     3$              ;BR IF AVAILABLE
4212  017044  030067  162452                BIT     R0,     MEMMAP  ;CHECK FOR BANK EXISTS.
4213  017050  001764                        BEQ     2$              ;BR IF NO BANK FOUND.
4214  017052  013737  172346  172344  3$:   MOV     @#KIPAR3,@#KIPAR2 ;COPY PAR
4215  017060  010046                        MOV     R0,-(SP)        ;;PUSH R0 ON STACK
4216  017062  010146                        MOV     R1,-(SP)        ;;PUSH R1 ON STACK
4217  017064  162737  000200  172344  4$:   SUB     #200,   @#KIPAR2 ;BACK DOWN WITH LOW PAR.
4218  017072  006001                        ROR     R1              ;SHIFT POINTER.
4219  017074  006000                        ROR     R0              ;...LO 64K.
4220  017076  103457                        BCS     90$             ;BR IF OVERFLOW.
4221  017100  030167  162420          5$:   BIT     R1,     MEMMAP+2 ;      CHECK IF BANK EXISTS...HI 64K.
4222  017104  001003                        BNE     6$              ;BR IF BANK EXISTS.
4223  017106  030067  162410                BIT     R0,     MEMMAP  ;CHECK IF BANK EXISTS...LO 64K.
4224  017112  001764                        BEQ     4$              ;BR IF BANK DOESN'T EXIST.
4225  017114  052601                  6$:   BIS     (SP)+,  R1      ;GET SECOND BANK POINTER.
4226  017116  052600                        BIS     (SP)+,  R0      ;...LO 64K.
4227  017120  030067  161456                BIT     R0,     PRGMAP  ;CHECK FOR CONFLICT.
4228  017124  001044                        BNE     90$             ;ABORT IF DESTINATION OVERLAYS SOURCE.
4229  017126  004567  177506                JSR     R5,     RELOC   ;GO RELOCATE PROGRAM.
4230  017132  000000                        .WORD   0               ;SOURCE FIRST ADDRESS.
4231  017134  040000                        .WORD   40000           ;DESTINATION FIRST ADDRESS.
4232  017136  013737  172344  172340        MOV     @#KIPAR2,@#KIPAR0 ;RELOCATE LO BANK
4233  017144  013737  172346  172342        MOV     @#KIPAR3,@#KIPAR1 ;RELOCATE HI BANK.
4234                        ;* PROGRAM SHOULD NOW BE EXICUTING OUT OF LAST TWO BANKS OF MEMORY.
4235  017152  010167  161426                MOV     R1,     PRGMAP+2 ;RESET PROGRAM MAP.
4236  017156  000473                        BR      30$             ;BR TO COMMON EXIT.
4237
4238  017160  012700  000400          10$:  MOV     #BIT8,  R0      ;SET BANK POINTER TO TOP OF MEM.
4239  017164  005001                        CLR     R1              ;SET ADDRESS POINTER TO TOP.
4240  017166  162701  020000          11$:  SUB     #20000, R1      ;BACK DOWN ONE BANK.
4241  017172  006200                        ASR     R0              ;MOVE POINTER DOWN ONE BANK.
4242  017174  103420                        BCS     90$             ;BR IF OVERFLOW.
4243  017176  030067  162320                BIT     R0,     MEMMAP  ;CHECK IF THIS BANK EXISTS.
4244  017202  001771                        BEQ     11$             ;BR IF NON-EXISTANT BANK.
4245  017204  162701  020000                SUB     #20000, R1      ;BACK DOWN TO NEXT BANK.
4246  017210  006200                        ASR     R0              ;MOV POINTER DOWN ONE BANK.
4247  017212  103411                        BCS     90$             ;BR IF OVERFLOW.
4248  017214  030067  162302                BIT     R0,     MEMMAP  ;CHECK IF THIS BANK EXISTS.
4249  017220  001762                        BEQ     11$             ;BR TO START OVER IF NO LOWER BANK.
4250  017222  010046                        MOV     R0,     -(SP)   ;SAVE THE POINTER.
4251  017224  006300                        ASL     R0              ;RESET POINTER TO HI BANK.
4252  017226  052600                        BIS     (SP)+,  R0      ;SET BIT FOR LO BANK.
4253  017230  030067  161346                BIT     R0,     PRGMAP  ;CHECK FOR A PROGRAM CONFLICT.
4254  017234  001401                        BEQ     12$             ;BR IF NO CONFLICT.
4255  017236                          90$:
4256  017236  000000                        HALT                    ;FATAL ERROR!!!  NOT ENOUGH MEMORY??
4257  017240  010167  000006          12$:  MOV     R1,     13$     ;SET DATA FOR RELOCATION SUBROUTINE.
4258  017244  004567  177370                JSR     R5,     RELOC   ;GO RELOCATE THE PROGRAM TO TOP OF MEM.
4259  017250  000000                        .WORD   0               ;SOURCE STARTING ADDRESS.
4260  017252  000000                  13$:  .WORD   0               ;DESTINATION STARTING ADDRESS.
4261  017254  010167  161320                MOV     R1,     RELOCF  ;SET RELOCATION FACTOR IN UNRELOCATED CODE.
4262  017260  060107                        ADD     R1,     PC      ;JUMP TO RELOCATED PROGRAM
4263                        ;*  PROGRAM NOW EXICUTING OUT OF TOP OF MEMORY.
4264  017262  060106                        ADD     R1,     SP      ;ADJUST THE STACK POINTER TO TOP OF MEMORY.
4265  017264  010167  161310                MOV     R1,     RELOCF  ;SET THE RELOCATION FACTOR.
4266  017270  060137  000004                ADD     R1,     @#ERRVEC ;ADJUST ERROR VECTOR.
```

```
4267  017274  060137  000024                ADD     R1,     @#PWRVEC ;ADJUST POWER FAIL VECTOR.
4268  017300  060137  000114                ADD     R1,     @#PARVEC ;ADJUST PARITY ERROR VECTOR.
4269  017304  026727  161626  177570        CMP     SWR,    #177570 ;CHECK FOR HARDWARE SWITCH REGISTER.
4270  017312  001404                        BEQ     14$             ;BR IF HARDWARE SWITCH REGISTER.
4271  017314  060167  161616                ADD     R1,     SWR     ;ADJUST SOFTWARE SWITCH REGISTER.
4272  017320  060167  161614                ADD     R1,     DISPLAY ;ADJUST SOFTWARE DISPLAY REGISTER.
4273  017324  062701  001610        14$:    ADD     #RADTAB,R1      ;POINT TO THE RELATIVE RELOCATION TABLE.
4274  017330  066721  161244        15$:    ADD     RELOCF, (R1)+   ;ADD RELOCATION FACTOR TO ADDRESSES IN TABLE.
4275  017334  005721                16$:    TST     (R1)+           ;CHECK FOR INTERUM TERMINATOR.
4276  017336  001776                        BEQ     16$             ;BR SO AS TO NOT MODIFY ZERO.
4277  017340  024127  177777                CMP     -(R1),  #-1     ;CHECK FOR END OF TABLE.
4278  017344  001371                        BNE     15$             ;BR IF MORE IN TABLE.
4279  017346  010067  161230        30$:    MOV     R0,     PRGMAP  ;SET NEW PROGRAM MAP...LO 64K.
4280  017352  012601                        MOV     (SP)+,R1        ;;POP STACK INTO R1
4281  017354  012600                        MOV     (SP)+,R0        ;;POP STACK INTO R0
4282  017356  066716  161216                ADD     RELOCF, (SP)    ;ADJUST RETURN ADDRESS.
4283  017362  000207                        RTS     PC              ;RETURN
4284
4285                                ;*****************************************************************
4286                                ;*  SUBROUTINE TO RELOCATE PROGRAM BACK TO BANKS 0 AND 1.
4287                                ;*****************************************************************
4288  017364  032767  000003  161210 RELO:    BIT     #3,     PRGMAP  ;CHECK FOR PROGRAM ALREADY IN BANKS 0 OR 1.
4289  017372  001401                        BEQ     1$              ;BR IF NO CONFLICT.
4290  017374  000000                        HALT                    ;FATAL ERROR!!!  PROGRAM ALREADY IN BANKS 0 OR 1!!!!
4291  017376  005767  161204        1$:     TST     MMAVA           ;CHECK FOR MEM MGMT.
4292  017402  001417                        BEQ     10$             ;BR IF NO MEMMGMT.
4293  017404  005037  172344                CLR     @#KIPAR2        ;SET PAR 2 TO BANK 0.
4294  017410  012737  000200  172346        MOV     #200,   @#KIPAR3 ;SET PAR 3 TO BANK 1.
4295  017416  004567  177216                JSR     R5,     RELOC   ;GO MOVE 8K INTO BANKS 0 AND 1.
4296  017422  000000                        .WORD   0               ;SOURCE STARTING ADDRESS.
4297  017424  040000                        .WORD   40000           ;DESTINATION STARTING ADDRESS.
4298  017426  005037  172340                CLR     @#KIPAR0        ;RESTORE PAR 0 TO BANK0.
4299  017432  012737  000200  172342        MOV     #200,   @#KIPAR1 ;RESTORE PAR 1 TO BANK 1.
4300                                ;*  PROGRAM IS NOW EXICUTING OUT OF BANKS 0 AND 1.
4301  017440  000444                        BR      30$             ;BR TO COMMON EXIT.
4302
4303  017442  016746  161132        10$:    MOV     RELOCF, -(SP)   ;PUT RELOCATION FACTOR ONTO THE STACK.
4304  017446  011667  000004                MOV     (SP),   20$     ;SET DATA FOR RELOC SUBROUTINE.
4305  017452  004567  177162                JSR     R5,     RELOC   ;GO MOVE THE PROGRAM BACK TO BANKS 0 AND 1.
4306  017456  000000                20$:    .WORD   0               ;SOURCE STARTING ADDRESS.
4307  017460  000000                        .WORD   0               ;DESTINATION STARTING ADDRESS.
4308  017462  161607                        SUB     (SP),   PC      ;JUMP TO RELOCATED PROGRAM.
4309                                ;*  THE PROGRAM IS NOW EXICUTING OUT OF BANKS 0 AND 1.
4310  017464  161606                        SUB     (SP),   SP      ;RESET THE STACK POINTER.
4311  017466  010046                        MOV     R0,-(SP)        ;;PUSH R0 ON STACK
4312  017470  012700  001610                MOV     #RADTAB,R0      ;SET UP POINTER TO RELATIVE ADDRESS TABLE.
4313  017474  166620  000002        21$:    SUB     2(SP),  (R0)+   ;RESET ADDRESSES TO UNRELOCATED VALUES.
4314  017500  005720                22$:    TST     (R0)+           ;CHECK FOR TERMINATORS.
4315  017502  001776                        BEQ     22$             ;BR OVER TERMINATORS.
4316  017504  024027  177777                CMP     -(R0),  #-1     ;CHECK FOR END OF TABLE INDICATOR.
4317  017510  001371                        BNE     21$             ;BR IF MORE ADDRESSES IN TABLE.
4318  017512  012600                        MOV     (SP)+,R0        ;;POP STACK INTO R0
4319  017514  161637  000004                SUB     (SP),   @#ERRVEC ;ADJUST ERROR VECTOR.
4320  017520  161637  000024                SUB     (SP),   @#PWRVEC ;ADJUST POWER FAIL VECTOR.
4321  017524  161637  000114                SUB     (SP),   @#PARVEC ;ADJUST PARITY ERROR VECTOR.
4322  017530  026727  161402  177570        CMP     SWR,    #177570 ;CHECK FOR HARDWARE SWITCH REGISTER.
```

```
 4323  017536  001404                        BEQ     23$             ;BR IF HARDWARE SWITCH REGISTER.
 4324  017540  161667  161372                SUB     (SP),   SWR     ;ADJUST SOFTWARE SWITCH REGISTER.
 4325  017544  161667  161370                SUB     (SP),   DISPLAY ;ADJUST SOFTWARE DISPLAY REGISTER.
 4326  017550  162616              23$:      SUB     (SP)+,  (SP)    ;ADJUST RETURN ADDRESS.
 4327  017552  005067  161022      30$:      CLR     RELOCF          ;RESET RELOCATION FACTOR.
 4328  017556  012767  000003  161016        MOV     #3,     PRGMAP  ;SET PROGRAM MAP TO POINT TO BANKS 0 AND 1.
 4329  017564  005067  161014                CLR     PRGMAP+2        ;...HI 64K.
 4330  017570  000207                        RTS     PC              ;RETURN.
 4331
 4332                              ;*****************************************************************
 4333                              ;* THIS SUBROUTINE MOVES THE LOADER AREA BACK TO THE "TOP" OF MEMORY FROM
 4334                              ;*  WHENCE IT CAME.  THE LOADER AREA IS SAVED AT THE END OF THE 8K OF
 4335                              ;*  PROGRAM CODE WHEN THE PROGRAM IS INITIALLY RUN.
 4336                              ;*****************************************************************
 4337  017572  016700  161720      RESLDR: MOV     LMAD,   R0      ;CHECK IF THE LOADERS WERE SAVED.
 4338  017576  001001                        BNE     1$              ;BR IF LOADER AREA WAS SAVED.
 4339  017600  000000                        HALT                    ;FATAL ERROR!!!  CAN'T RESTORE LOADER AREA IF IT WASN'T SAVED.
 4340  017602  005767  161000      1$:       TST     MMAVA           ;CHECK FOR MEM MGMT.
 4341  017606  001402                        BEQ     2$              ;SKIP IF NO MEM MGMT.
 4342  017610  005037  177572                CLR     @#SR0           ;DISABLE MEM MGMT.
 4343  017614  012701  040000      2$:       MOV     #40000, R1      ;GET END OF 8K, ASSUME PROG NOT RELOCATED.
 4344  017620  012702  002734                MOV     #1500., R2      ;GET COUNTER.
 4345  017624  014140              3$:       MOV     -(R1),  -(R0)   ;MOVE THE LOADER AREA.
 4346  017626  005302                        DEC     R2              ;COUNT HOW LONG THE AREA IS.
 4347  017630  001375                        BNE     3$              ;BR IF NOT MORE TO MOVE.
 4348  017632  005767  160750                TST     MMAVA           ;CHECK FOR MEM MGMT.
 4349  017636  001402                        BEQ     4$              ;BR IF NO MEM MGMT.
 4350  017640  005237  177572                INC     @#SR0           ;ENABLE MEM MGMT.
 4351  017644  000207              4$:       RTS     PC              ;RETURN.
```

```
                                        .SBTTL  PARITY MEMORY TRAP SERVICE AND SUBROUTINES.
                                        ;*****************************************************************
                                        ;* PARITY MEMORY UNEXPECTED ERROR TRAP SERVICE ROUTINE.
                                        ;*  FIND OUT WHICH REGISTER DETECTED THE ERROR.
                                        ;*  THEN SCAN MEMORY TO SEE IF PARITY ERROR STILL SET AND REPORT LOCATION.
                                        ;*****************************************************************
017646  011667  161250          PESRV:  MOV     (SP),   $BDADR  ;GET PC OF INSTRUCTION WHICH CAUSED ERROR.
017652  004567  003362                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
017656  026060                          .WORD   UNEXPT          ;ADDRESS OF MESSAGE TO BE TYPED
                                                                ;"UNEXPECTED MEMORY PARITY TRAP."
017660  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
017662  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
017664  016703  161726                  MOV     .MPRX,  R3      ;GET POINTER TO PARITY REGISTERS.
017670  005733                  1$:     TST     @(R3)+          ;CHECK THE PARITY REG FOR AN ERROR FLAG.
017672  100415                          BMI     3$              ;BR IF THIS REGISTER SHOWS THE ERROR.
017674  005713                          TST     (R3)            ;CHECK FOR TABLE TERMINATOR.
017676  001374                          BNE     1$              ;BR IF MORE REGISTERS.
017700  004767  002132                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
                                ;***ERROR***    NO REGISTER INDICATED ERROR
017704  000024                          .WORD   24              ;ERROR TYPE CODE.
017706  000417                          BR      4$              ;EXIT
017710  005713                  2$:     TST     (R3)            ;CHECK FOR TABLE TERMINATOR.
017712  001415                          BEQ     4$              ;BR IF NO MORE PARITY REGISTERS.
017714  005733                          TST     @(R3)+          ;CHECK THE PARITY REG FOR AN ERROR FLAG.
017716  100374                          BPL     2$              ;BR IF NO ERROR FLAG.
017720  004567  003314                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
017724  026151                          .WORD   MTOE            ;ADDRESS OF MESSAGE TO BE TYPED
                                                                ;"MORE THAN ONE ERROR FOUND."
017726                          3$:
017726  004767  000610          64$:    JSR     PC,     $PRNTQ  ;SET UP VALUES FOR ERROR PRINTING.
017732  004767  002100                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
017736  000025                          .WORD   25              ;ERROR TYPE CODE.
017740  004767  000216                  JSR     PC,     PSCAN   ;GO SCAN MEMORY FOR BAD PARITY.
017744  000761                          BR      2$              ;GO LOOK FOR MORE ERRORS.
017746                          4$:
017746  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
017750  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
017752  000002                          RTI                     ;RETURN.

                                        ;*****************************************************************
                                        ;ROUTINE TO ENABLE PARITY ERROR ACTION ON MA/MF PARITY MEMORIES
                                        ;THIS ROUTINE IS MEANT TO CATCH UNEXPECTEDS
                                        ;*****************************************************************
017754  005767  162272          MAMF:   TST     MPRX            ;CHECK IF ANY PARITY REGISTERS EXIST.
017760  001434                          BEQ     MAMF2           ;EXIT IF NO PARITY REGISTERS.
017762  032777  000100  161146          BIT     #SW6,   @SWR    ;CHECK FOR INHIBIT PARITY ERROR DETECTION.
017770  001030                          BNE     MAMF2           ;EXIT IF NO PARITY ERROR DETECTION.
017772  005767  160602                  TST     RELOCF          ;CHECK IF PROGRAM RELOCATED OUT OF BANK 0.
017776  001410                          BEQ     SETAE           ;BR IF PROG IN BANK 0.
020000  032777  000040  161130          BIT     #SW5,   @SWR    ;CHECK IF VECTORS PROTECTED.
020006  001004                          BNE     SETAE           ;BR IF VECTOR AREA PROTECTED.
020010  026727  161544  001000          CMP     FSTADR, #1000   ;CHECK FOR STARTING ADDRESS ABOVE THE VECTORS.
020016  103415                          BLO     MAMF2           ;EXIT IF VECTORS EXPOSED TO TESTING.
020020  016737  161606  000114  SETAE:  MOV     .PESRV, @#PARVEC ;SET PARITY ERROR TRAP VECTOR
020026  005037  000116                  CLR     @#PARVEC+2      ;PRIORITY LEVEL 0 ON TRAP
020032  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
```

```
 4408  020034  016703  161556          MAMF1:  MOV     .MPRX,  R3      ;GET PARITY REGISTER TABLE POINTER.
 4409  020040  052733  000001                  BIS     #AE,    @(R3)+  ;SET ACTION ENABLE BIT IN PARITY REG
 4410  020044  005713                          TST     (R3)            ;CHECK FOR END OF TABLE.
 4411  020046  001374                          BNE     MAMF1           ;BR IF MORE PARITY REGISTERS.
 4412  020050  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
 4413  020052  000207                  MAMF2:  RTS     PC              ;RETURN.
 4414
 4415                                  ;*****************************************************************
 4416                                  ;*  SUBROUTINE TO CHECK PARITY REGISTERS FOR ERRORS THAT DIDN'T TRAP.
 4417                                  ;*****************************************************************
 4418  020054  005767  162172          CKPMER: TST     MPRX            ;CHECK IF ANY PARITY REGISTERS EXIST.
 4419  020060  001437                          BEQ     4$              ;BR IF NO PARITY REGISTERS.
 4420  020062  032777  000100  161046          BIT     #SW6,   @SWR    ;CHECK FOR INHIBIT PARITY ERROR CHECKING.
 4421  020070  001033                          BNE     4$              ;BR IF PARITY ERROR CHECKING INHIBITED.
 4422  020072  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
 4423  020074  016703  161516                  MOV     .MPRX,  R3      ;GET PARITY REG TABLE POINTER.
 4424  020100  005733                  1$:     TST     @(R3)+          ;CHECK THE PARITY REG FOR AN ERROR FLAG.
 4425  020102  100023                          BPL     3$              ;BR IF NO ERROR
 4426  020104  032773  000001  177776          BIT     #BIT0,  @-2(R3) ;CHECK IF A TRAP SHOULD HAVE OCCURRED.
 4427  020112  001010                          BNE     2$              ;BR IF NO ACTION ENABLE.
 4428  020114  004767  000422          64$:    JSR     PC,     SPRNTQ  ;SET UP VALUES FOR ERROR PRINTING.
 4429  020120  004767  001712                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
 4430  020124  000026                          .WORD   26              ;ERROR TYPE CODE.
 4431  020126  000411                          BR      3$
 4432  020130  004767  000026                  JSR     PC,     PSCAN   ;GO SCAN ALL MEMORY FOR PARITY ERRORS.
 4433  020134                          2$:
 4434  020134  004767  000402          65$:    JSR     PC,     SPRNTQ  ;SET UP VALUES FOR ERROR PRINTING.
 4435  020140  004767  001672                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
 4436  020144  000027                          .WORD   27              ;ERROR TYPE CODE.
 4437  020146  004767  000010                  JSR     PC,     PSCAN   ;GO SCAN ALL MEMORY FOR PARITY ERRORS.
 4438  020152  005713                  3$:     TST     (R3)            ;CHECK FOR TABLE TERMINATOR.
 4439  020154  001351                          BNE     1$              ;BR IF MORE.
 4440  020156  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
 4441  020160  000207                  4$:     RTS     PC              ;RETURN.
 4442
 4443                                  ;*****************************************************************
 4444                                  ;*  THIS SUBROUTINE WILL SCAN ALL OF MEMORY LOOKING FOR BAD PARITY,
 4445                                  ;*  TYPE OUT ALL LOCATIONS FOUND TO BE BAD, AND WRITE BACK INTO THE
 4446                                  ;*  LOCATIONS IN ORDER TO RESTORE GOOD PARITY.
 4447                                  ;*****************************************************************
 4448  020162                          PSCAN:
 4449  020162  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
 4450  020164  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
 4451  020166  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
 4452  020170  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
 4453  020172  010446                          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
 4454  020174  013746  000114                  MOV     @#114,-(SP)     ;;PUSH @#114 ON STACK
 4455  020200  013746  000116                  MOV     @#116,-(SP)     ;;PUSH @#116 ON STACK
 4456  020204  004567  003030                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 4457  020210  026215                          .WORD   SCANM           ;ADDRESS OF MESSAGE TO BE TYPED
 4458                                                                  ;"SCANNING MEMORY FOR BAD PARITY."
 4459  020212  012700  000001                  MOV     #BIT0,  R0      ;SET BIT POINTER TO FIRST BANK.
 4460  020216  005001                          CLR     R1              ;CLR HI 64K POINTER.
 4461  020220  005002                          CLR     R2              ;INIT ADDRESS POINTER.
 4462  020222  005004                          CLR     R4              ;INIT ERROR DETECTED FLAG.
 4463  020224  004767  000256                  JSR     PC,     CLRPAR  ;CLEAR THE PARITY REGISTERS.
```

```
 4464  020230  012737  000116  000114          MOV     #116,   @#114   ;HALT IF ANOTHER PARITY TRAP.
 4465  020236  005037  000116                  CLR     @#116
 4466  020242  005767  160340                  TST     MMAVA           ;CHECK FOR MEMORY MANAGEMENT.
 4467  020246  001406                          BEQ     1$              ;BR IF NO MEM MGMT.
 4468  020250  013746  172344                  MOV     @#KIPAR2,-(SP)          ;;PUSH @#KIPAR2 ON STACK
 4469  020254  005037  172344                  CLR     @#KIPAR2        ;INIT MEM MGMT TO POINT TO BANK 0.
 4470  020260  012702  040000                  MOV     #40000, R2      ;SET ADR POINTER TO PAR2.
 4471  020264  030067  161232          1$:     BIT     R0,     MEMMAP  ;CHECK IF THIS BANK OF MEM EXISTS.
 4472  020270  001003                          BNE     2$              ;BR IF THIS BANK EXISTS.
 4473  020272  030167  161226                  BIT     R1,     MEMMAP+2 ;CHECK HI 64K MAP.
 4474  020276  001442                          BEQ     10$             ;BR IF THIS BANK DOESN'T EXIST.
 4475  020300                          2$:
 4476  020300  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
 4477  020302  111201                  3$:     MOVB    (R2),   R1      ;READ THE LOCATION TO SEE IF IT HAS A PARITY ERROR.
 4478  020304  016703  161306                  MOV     .MPRX,  R3      ;SET UP POINTER TO PARITY REGISTERS.
 4479  020310  005733                  4$:     TST     @(R3)+          ;CHECK FOR THE ERROR FLAG.
 4480  020312  100024                          BPL     6$              ;BR IF NO ERROR FLAG.
 4481  020314  005704                          TST     R4              ;CHECK IF FIRST ERROR, THIS SCAN.
 4482  020316  001003                          BNE     5$              ;BR IF MORE THAN ONE ERROR FOUND.
 4483  020320  005367  160566                  DEC     $ERTTL          ;ADJUST ERROR COUNT.
 4484  020324  005204                          INC     R4              ;SET FLAG TO INDICATE ERROR FOUND.
 4485  020326                          5$:
 4486  020326  004767  000210          64$:    JSR     PC,     SPRNTQ  ;SET UP VALUES FOR ERROR PRINTING.
 4487  020332  004767  001500                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
 4488  020336  000030                          .WORD   30              ;ERROR TYPE CODE.
 4489  020340  111212                          MOVB    (R2),   (R2)    ;REWRITE THE LOCATION TO CLEAR BAD PARITY.
 4490  020342  005053                          CLR     @-(R3)          ;CLEAR THE ERROR FLAG.
 4491  020344  105712                          TSTB    (R2)            ;CHECK IF THE PARITY ERROR WAS CLEARED.
 4492  020346  005733                          TST     @(R3)+          ;CHECK FOR THE ERROR FLAG.
 4493  020350  100005                          BPL     6$              ;BR IF IT IS OK.
 4494  020352  004567  002662                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 4495  020356  026257                          .WORD   PEWNC           ;ADDRESS OF MESSAGE TO BE TYPED
 4496                                                                  ;"PARITY ERROR WILL NOT CLEAR."
 4497  020360  005073  177776                  CLR     @-2(R3)         ;CLEAR OUT THE PARITY ERROR FLAG.
 4498  020364  005713                  6$:     TST     (R3)            ;CHECK FOR THE END OF REG ADR TABLE.
 4499  020366  001350                          BNE     4$              ;BR IF MORE PARITY REGISTERS.
 4500  020370  005202                          INC     R2              ;GO TO NEXT MEMORY ADDRESS.
 4501  020372  032702  017777                  BIT     #MASK4K,R2      ;CHECK FOR END OF 4K BANK.
 4502  020376  001341                          BNE     3$              ;BR IF MORE MEMORY THIS BANK.
 4503  020400  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
 4504  020402  000402                          BR      11$             ;BR TO CHECK FOR NEXT BANK.
 4505  020404  062702  020000          10$:    ADD     #20000, R2      ;SKIP BANKS THAT AREN'T THERE.
 4506  020410  005767  160172          11$:    TST     MMAVA           ;CHECK FOR MEM MGMT.
 4507  020414  001413                          BEQ     12$             ;BR IF NO MEM MGMT.
 4508  020416  062737  000200  172344          ADD     #200,   @#KIPAR2 ;UPDATE MEM MGMT REG TO NEXT 4K.
 4509  020424  012702  040000                  MOV     #40000, R2      ;RESET ADDRESS POINTER TO BEGINNING OF BANK.
 4510  020430  006300                          ASL     R0              ;UPDATE BANK POINTER.
 4511  020432  006101                          ROL     R1              ;...HI 64K.
 4512  020434  100313                          BPL     1$              ;BR IF MORE BANKS.
 4513  020436  012637  172344                  MOV     (SP)+,@#KIPAR2          ;;POP STACK INTO @#KIPAR2
 4514  020442  000402                          BR      20$             ;GO CHECK IF ANY ERRORS FOUND.
 4515  020444  106300                  12$:    ASLB    R0              ;UPDATE POINTER TO NEXT BANK.
 4516  020446  100306                          BPL     1$              ;BR IF MORE BANKS.
 4517  020450  005704                  20$:    TST     R4              ;CHECK IF ANY PARITY ERRORS DETECTED.
 4518  020452  001003                          BNE     21$             ;BR IF ERRORS DETECTED.
 4519  020454  004567  002560                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
```

```
4520  020460  025430                          .WORD   NOPES           ;ADDRESS OF MESSAGE TO BE TYPED
4521  020462                        21$:
4522  020462  012637  000116                  MOV     (SP)+,@#116     ;;POP STACK INTO @#116
4523  020466  012637  000114                  MOV     (SP)+,@#114     ;;POP STACK INTO @#114
4524  020472  012604                          MOV     (SP)+,R4        ;;POP STACK INTO R4
4525  020474  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
4526  020476  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
4527  020500  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
4528  020502  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
4529  020504  000207                          RTS     PC              ;RETURN.
4530
4531                                  ;*****************************************************************
4532                                  ;ROUTINE TO CLEAR ALL PARITY REGISTERS PRESENT
4533                                  ;*****************************************************************
4534  020506                        CLRPAR:
4535  020506  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
4536  020510  016703  161102                  MOV     .MPRX,  R3      ;GET PARITY REGISTER TABLE POINTER.
4537  020514  005713                1$:       TST     (R3)            ;CHECK FOR THE TABLE TERMINATOR.
4538  020516  001402                          BEQ     2$              ;BR IF DONE ALL PARITY REGISTERS.
4539  020520  005033                          CLR     @(R3)+          ;CLEAR THE PARITY REGISTER.
4540  020522  000774                          BR      1$              ;BR FOR MORE
4541  020524                        2$:
4542  020524  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
4543  020526  000207                          RTS     PC              ;RETURN.
4544
4545                                  .SBTTL  SUBROUTINES TO SET UP DATA FOR ERROR PRINTOUT ROUTINE.
4546                                  ;*****************************************************************
4547                                  ;*  THESE ROUTINES ARE USED TO TRANSFER DATA TO COMMON TAG AREA (.$CMTAG)
4548                                  ;*  FOR ERROR PRINTOUT BY .$ERROR & .$ERRTYP ROUTINES FROM **SYSMAC**.
4549                                  ;*****************************************************************
4550  020530  010267  160364        SPRNT:  MOV     R2,     $GDADR  ;SAVE THE ADDRESS UNDER TEST.
4551  020534  005067  160364                CLR     $GDDAT          ;SHOULD BE DATA IS "0".
4552  020540  000430                        BR      SPRNTB
4553
4554  020542  014367  160410        SPRNTQ: MOV     -(R3),  $TMP0   ;GET THE PARITY REGISTER ADDRESS.
4555  020546  013367  160406                MOV     @(R3)+, $TMP1   ;GET THE CONTENTS OF THE PARITY REG.
4556  020552  000402                        BR      SPRNTO
4557
4558  020554  011367  160376        SPRNTP: MOV     (R3),   $TMP0   ;GET THE PARITY REGISTER ADDRESS.
4559  020560  010267  160334        SPRNTO: MOV     R2,     $GDADR  ;GET THE MEMORY ADDRESS BEING TESTED
4560  020564  000414                        BR      SPRNTA          ;BR TO COMMON SECTION.
4561
4562  020566  010267  160326        SPRNT1: MOV     R2,     $GDADR  ;GET THE MEMORY ADDRESS BEING TESTED
4563  020572  005367  160322                DEC     $GDADR          ;ADJUST IT FOR PRINTOUT.
4564  020576  000407                        BR      SPRNTA          ;BR TO COMMON SECTION.
4565
4566  020600  010367  160352        SPRNT3: MOV     R3,     $TMP0   ;GET THE DATA IN R3.
4567  020604  010267  160310        SPRNT2: MOV     R2,     $GDADR  ;GET THE MEMORY ADDRESS BEING TESTED
4568  020610  162767  000002  160302        SUB     #2,     $GDADR  ;ADJUST IT FOR PRINTOUT.
4569  020616  010067  160302        SPRNTA: MOV     R0,     $GDDAT  ;GET WHAT THE DATA SHOULD BE
4570  020622  010167  160300        SPRNTB: MOV     R1,     $BDDAT  ;GET WHAT THE DATA WAS
4571  020626  000207                        RTS     PC              ;RETURN TO ENTER ERROR ROUTINES
4572
4573                                  ;*****************************************************************
4574                                  ;*  SUBROUTINE TO TYPE OUT A MAP OF 4K BANK.
4575                                  ;*  R0 POINTS TO THE MAP UPON ENTERING THIS ROUTINE.
```

```
4576                                    ;*****************************************************************
4577  020630  005710                    TYPMAP: TST     (R0)            ;CHECK IF ANY MEMORY IN MAP...LO 64K.
4578  020632  001007                            BNE     1$              ;BR IF MEMORY IN MAP.
4579  020634  005760  000002                    TST     2(R0)           ;...HI 64K.
4580  020640  001004                            BNE     1$              ;BR IF MEMORY IN MAP.
4581  020642  004567  002372                    JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
4582  020646  025645                            .WORD   NOMEM           ;ADDRESS OF MESSAGE TO BE TYPED
4583                                                                    ;"NO MEMORY FOUND."
4584  020650  000475                            BR      6$              ;EXIT
4585  020652                          1$:
4586  020652  010146                            MOV     R1,-(SP)        ;;PUSH R1 ON STACK
4587  020654  010246                            MOV     R2,-(SP)        ;;PUSH R2 ON STACK
4588  020656  010346                            MOV     R3,-(SP)        ;;PUSH R3 ON STACK
4589  020660  010446                            MOV     R4,-(SP)        ;;PUSH R4 ON STACK
4590  020662  012701  000001                    MOV     #BIT0,  R1      ;SET UP BANK POINTER...LO 64K.
4591  020666  005002                            CLR     R2              ;...HI 64K.
4592  020670  012703  177777                    MOV     #-1,    R3      ;SET UP ADDRESS POINTER TO -1.
4593  020674  010304                            MOV     R3,     R4      ;HI BITS OF ADDRESS AS WILL.
4594  020676  030110                  2$:       BIT     R1,     (R0)    ;CHECK THE MAP FOR THIS BANK.
4595  020700  001014                            BNE     3$              ;BR IF THIS BANK PRESENT.
4596  020702  030260  000002                    BIT     R2,     2(R0)   ;CHECK HI 64K MAP.
4597  020706  001011                            BNE     3$              ;BR IF THIS BANK PRESENT.
4598  020710  105703                            TSTB    R3              ;CHECK FOR PREVIOUS PRINTOUT.
4599  020712  001042                            BNE     5$              ;BR IF ALREADY TYPED "TO"
4600  020714  162703  000001                    SUB     #1,     R3      ;BACK UP TO LAST ADR OF PREVIOUS BANK.
4601  020720  005604                            SBC     R4              ;...HI ADDRESS BITS.
4602  020722  004567  002312                    JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
4603  020726  025241                            .WORD   TO              ;ADDRESS OF MESSAGE TO BE TYPED
4604  020730  000410                            BR      4$              ;GO TO TYPE THE ADDRESS.
4605  020732  105703                  3$:       TSTB    R3              ;CHECK FOR PREVIOUS TYPEOUT.
4606  020734  001431                            BEQ     5$              ;BR IF ALREADY TYPE "FROM".
4607  020736  062703  000001                    ADD     #1,     R3      ;POINT TO FIRST ADDRESS OF THIS BANK.
4608  020742  005504                            ADC     R4              ;...HI BITS OF ADDRESS.
4609  020744  004567  002270                    JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
4610  020750  025231                            .WORD   FROM            ;ADDRESS OF MESSAGE TO BE TYPED
4611  020752                          4$:
4612  020752  010346                            MOV     R3,-(SP)        ;;PUSH R3 ON STACK
4613  020754  010446                            MOV     R4,-(SP)        ;;PUSH R4 ON STACK
4614  020756  006303                            ASL     R3              ;BIT 15 INTO C-BIT
4615  020760  006104                            ROL     R4              ;BIT 15 INTO R4.
4616  020762  006003                            ROR     R3              ;RESTORE BITS 14-0.
4617  020764  010446                            MOV     R4,-(SP)        ;;SAVE R4 FOR TYPEOUT
4618                                                                    ;;TYPE ADDRESS BITS 21-15
4619                                  ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOS ROUTINE
4620                                  ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
4621  020766  013746  177776                    MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
4622  020772  004767  003410                    JSR     PC,     $TYPOS  ;GO TO THE SUBROUTINE
4623  020776     003                            .BYTE   3               ;;TYPE 3 DIGIT(S)
4624  020777     000                            .BYTE   0               ;;SUPPRESS LEADING ZEROS
4625  021000  010346                            MOV     R3,-(SP)        ;;SAVE R3 FOR TYPEOUT
4626                                                                    ;;TYPE ADDRESS BITS 14-0
4627                                  ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOS ROUTINE
4628                                  ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
4629  021002  013746  177776                    MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
4630  021006  004767  003374                    JSR     PC,     $TYPOS  ;GO TO THE SUBROUTINE
4631  021012     005                            .BYTE   5               ;;TYPE 5 DIGIT(S)
```

```
4632  021013  001                          .BYTE   1               ;;TYPE LEADING ZEROS
4633  021014  012604                       MOV     (SP)+,R4        ;;POP STACK INTO R4
4634  021016  012603                       MOV     (SP)+,R3        ;;POP STACK INTO R3
4635  021020  062703  020000       5$:     ADD     #20000, R3      ;UPDATE TO NEXT BANK.
4636  021024  005504                       ADC     R4              ;...HI ADDRESS BITS.
4637  021026  006301                       ASL     R1              ;SHIFT POINTER...LO 64K.
4638  021030  006102                       ROL     R2              ;...HI 64K.
4639  021032  103321                       BCC     2$              ;BR IF MORE BANKS.
4640  021034  012604                       MOV     (SP)+,R4        ;;POP STACK INTO R4
4641  021036  012603                       MOV     (SP)+,R3        ;;POP STACK INTO R3
4642  021040  012602                       MOV     (SP)+,R2        ;;POP STACK INTO R2
4643  021042  012601                       MOV     (SP)+,R1        ;;POP STACK INTO R1
4644  021044  000207             6$:       RTS     PC              ;RETURN.
4645
4646                                ;*****************************************************************
4647
4648                                .SBTTL  SCOPE HANDLER ROUTINE
4649
4650                                ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
4651                                ;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
4652                                ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
4653                                ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
4654                                ;*SW14=1         LOOP ON TEST
4655                                ;*SW11=1         INHIBIT ITERATIONS
4656                                ;*SW09=1         LOOP ON ERROR
4657                                ;*SW08=1         LOOP ON TEST IN SWR<4:0>
4658                                ;*CALL
4659                                ;*      SCOPE           ;;SCOPE=IOT
4660
4661  021046                        $SCOPE:
4662  021046  012504                       MOV     (R5)+,  R4      ;SAVE MINIMUM BLOCK MASK NEXT TEST.
4663  021050  010516                       MOV     R5,     (SP)    ;PUT RETURN PC ONTO STACK, SIMULATE JSR PC.
4664  021052  032777  040000  160056 1$:     BIT     #BIT14,@SWR     ;;LOOP ON PRESENT TEST?
4665  021060  001117                       BNE     $OVER           ;;YES IF SW14=1
4666                                ;#####START OF CODE FOR THE XOR TESTER#####
4667  021062  000416             $XTSTR:   BR      6$              ;;IF RUNNING ON THE "XOR" TESTER CHANGE
4668                                                                ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
4669  021064  013746  000004               MOV     @#ERRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
4670  021070  012737  021110  000004       MOV     #5$,@#ERRVEC    ;;SET FOR TIMEOUT
4671  021076  005737  177060               TST     @#177060        ;;TIME OUT ON XOR?
4672  021102  012637  000004               MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
4673  021106  000466                       BR      $SVLAD          ;;GO TO THE NEXT TEST
4674  021110  022626             5$:       CMP     (SP)+,(SP)+     ;;CLEAR THE STACK AFTER A TIME OUT
4675  021112  012637  000004               MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
4676  021116  000426                       BR      7$              ;;LOOP ON THE PRESENT TEST
4677  021120                        6$:;#####END OF CODE FOR THE XOR TESTER#####
4678  021120  032777  000400  160010       BIT     #BIT08,@SWR     ;;LOOP ON SPEC. TEST?
4679  021126  001407                       BEQ     2$              ;;BR IF NO
4680  021130  017746  160002               MOV     @SWR,-(SP)      ;;SET DESIRED TEST NUM. FROM SWR
4681  021134  042716  000340               BIC     #$SWRMK,(SP)    ;;STRIP AWAY UNDESIRED BITS
4682  021140  122667  157736               CMPB    (SP)+,$TSTNM    ;;ON THE RIGHT TEST?
4683  021144  001465                       BEQ     $OVER           ;;BR IF YES
4684  021146  105767  157731       2$:     TSTB    $ERFLG          ;;HAS AN ERROR OCCURRED?
4685  021152  001421                       BEQ     3$              ;;BR IF NO
4686  021154  126767  157735  157721       CMPB    $ERMAX,$ERFLG   ;;MAX. ERRORS FOR THIS TEST OCCURRED?
4687  021162  101015                       BHI     3$              ;;BR IF NO
```

```
 4688  021164  032777  001000  157744          BIT   #BIT09,@SWR     ;;LOOP ON ERROR?
 4689  021172  001404                          BEQ   4$              ;;BR IF NO
 4690  021174  016767  157710  157704  7$:     MOV   $LPERR,$LPADR   ;;SET LOOP ADDRESS TO LAST SCOPE
 4691  021202  000446                          BR    $OVER
 4692  021204  105067  157673          4$:     CLRB  $ERFLG          ;;ZERO THE ERROR FLAG
 4693  021210  005067  157752                  CLR   $TIMES          ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
 4694  021214  000415                          BR    1$              ;;ESCAPE TO THE NEXT TEST
 4695  021216  032777  004000  157712  3$:     BIT   #BIT11,@SWR     ;;INHIBIT ITERATIONS?
 4696  021224  001011                          BNE   1$              ;;BR IF YES
 4697  021226  005767  157756                  TST   $PASS           ;;IF FIRST PASS OF PROGRAM
 4698  021232  001406                          BEQ   1$              ;;      INHIBIT ITERATIONS
 4699  021234  005267  157644                  INC   $ICNT           ;;INCREMENT ITERATION COUNT
 4700  021240  026767  157722  157636          CMP   $TIMES,$ICNT    ;;CHECK THE NUMBER OF ITERATIONS MADE
 4701  021246  002024                          BGE   $OVER           ;;BR IF MORE ITERATION REQUIRED
 4702  021250  012767  000001  157626  1$:     MOV   #1,$ICNT        ;;REINITIALIZE THE ITERATION COUNTER
 4703  021256  016767  000552  157702          MOV   $MXCNT,$TIMES   ;;SET NUMBER OF ITERATIONS TO DO
 4704  021264  105267  157612          $SVLAD: INCB  $TSTNM          ;;COUNT TEST NUMBERS
 4705  021270  116767  157606  157710          MOVB  $TSTNM,$TESTN   ;;SET TEST NUMBER IN APT MAILBOX
 4706  021276  011667  157604                  MOV   (SP),$LPADR     ;;SAVE SCOPE LOOP ADDRESS
 4707  021302  011667  157602                  MOV   (SP),$LPERR     ;;SAVE ERROR LOOP ADDRESS
 4708  021306  005067  157656                  CLR   $ESCAPE         ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
 4709  021312  112767  000001  157575          MOVB  #1,$ERMAX       ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
 4710  021320  016777  157556  157612  $OVER:  MOV   $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
 4711  021326  016716  157554                  MOV   $LPADR,(SP)     ;;FUDGE RETURN ADDRESS
 4712  021332  020516                  INSERT: CMP   R5,     (SP)    ;CHECK FOR LOOP ON TEST.
 4713  021334  001402                          BEQ   1$              ;BR IF START NEXT TEST.
 4714  021336  000167  000470                  JMP   ENDINS          ;JMP IF LOOP ON LAST TEST.
 4715  021342  012767  037777  160234  1$:     MOV   #37777, BLKMSK  ;SET 8K BOUNDRY MASK.
 4716  021350  005767  157634                  TST   $PASS           ;CHECK FOR PASS 0.
 4717  021354  001404                          BEQ   2$              ;BR IF PASS 0
 4718  021356  126727  157520  000023          CMPB  $TSTNM, #23     ;CHECK IF IN SECTION 3.
 4719  021364  103002                          BHIS  3$              ;BR IF IN SECTION 3.
 4720  021366  006267  160212          2$:     ASR   BLKMSK          ;RESET BOUNDRY TO 4K.
 4721  021372  016767  160162  160162  3$:     MOV   FSTADR, TMPFAD  ;GET FIRST ADDRESS.
 4722  021400  005767  157174                  TST   RELOCF          ;CHECK IF PRG RELOCATED.
 4723  021404  001430                          BEQ   4$              ;BR IF NOT RELOCATED.
 4724  021406  032777  000040  157522          BIT   #SW05,  @SWR    ;CHECK IF LOC 0-776 TO BE PROTECTED.
 4725  021414  001424                          BEQ   4$              ;BR IF SW NOT SET.
 4726  021416  026727  160140  001000          CMP   TMPFAD, #1000   ;CHECK IF NOT BEING TESTED.
 4727  021424  103020                          BHIS  4$              ;BR IF ALREADY PROTECTED.
 4728  021426  012767  001000  160126          MOV   #1000,  TMPFAD  ;RESET FIRST ADDRESS.
 4729  021434  052767  000001  160124          BIS   #BIT0,  FADMAP  ;SET FLAG IN FIRST BANK.
 4730  021442  026727  160124  001000          CMP   LSTADR, #1000   ;CHECK IF GONE PAST LAST ADR.
 4731  021450  101006                          BHI   4$              ;BR IF ENOUGH MEMORY.
 4732  021452  004567  001562                  JSR   R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 4733  021456  026316                          .WORD NOMTST          ;ADDRESS OF MESSAGE TO BE TYPED
 4734                                                                ;"NO MEMORY TESTED"
 4735  021460  016716  160154                  MOV   .TST32, (SP)    ;ADJUST RETURN ADR FOR ABORT.
 4736  021464  000207                          RTS   PC              ;ABORT.
 4737  021466  016767  160100  160100  4$:     MOV   LSTADR, TMPLAD  ;GET LAST ADDRESS.
 4738  021474  016767  160032  160024          MOV   SAVTST, TSTMAP  ;GET TEST MAP, LO 64K.
 4739  021502  016767  160026  160020          MOV   SAVTST+2,TSTMAP+2 ;...HI 64K.
 4740  021510  046767  157066  160010          BIC   PRGMAP, TSTMAP  ;DON'T TEST OVER THE PROGRAM.
 4741  021516  046767  157062  160004          BIC   PRGMAP+2,TSTMAP+2
 4742  021524  005767  157460                  TST   $PASS           ;CHECK FOR FIRST PASS
 4743  021530  001011                          BNE   10$             ;BR IF NOT FIRST PASS.
```

```
4744  021532  032767  000003  157766          BIT     #3,     TSTMAP  ;CHECK IF FIRST TWO BANKS AVAILABLE.
4745  021540  001405                          BEQ     10$             ;NOT TESTING FIRST 2 BANKS.
4746  021542  042767  177774  157756          BIC     #177774,TSTMAP  ;CLR ALL BUT FIRST 2 BANKS.
4747  021550  005067  157754                  CLR     TSTMAP+2
4748  021554  005704                  10$:    TST     R4              ;CHECK FOR A MINIMUM BLOCK SIZE.
4749  021556  001503                          BEQ     20$             ;BR IF NO MIN BLOCK SIZE.
4750  021560  030467  157776                  BIT     R4,     TMPFAD  ;CHECK IF FIRST ADR ON BLOCK BOUNDRY.
4751  021564  001416                          BEQ     11$             ;BR IF FIRST ADR ON BLOCK BOUNDRY.
4752  021566  050467  157770                  BIS     R4,     TMPFAD  ;ADJUST FIRST ADR TO END OF BLOCK.
4753  021572  005267  157764                  INC     TMPFAD          ;FIRST ADR TO FIRST ADR OF NEXT BLOCK.
4754  021576  032767  017777  157756          BIT     #MASK4K,TMPFAD  ;CHECK IF FIRST ADR REACHED 4K BOUNDRY.
4755  021604  001006                          BNE     11$             ;BR IF NOT ON 4K BOUNDRY.
4756  021606  046767  157754  157712          BIC     FADMAP, TSTMAP  ;DON'T TEST FIRST BANK.
4757  021614  046767  157750  157706          BIC     FADMAP+2,TSTMAP+2
4758  021622  030467  157746          11$:    BIT     R4,     TMPLAD  ;CHECK IF LAST ADR ON BLOCK BOUNDRY.
4759  021626  001414                          BEQ     12$             ;BR IF ON BLOCK BOUNDRY.
4760  021630  040467  157740                  BIC     R4,     TMPLAD  ;ADJUST LAST ADR DOWN TO NEXT BLOCK BOUNDRY.
4761  021634  032767  017777  157732          BIT     #MASK4K,TMPLAD  ;CHECK IF ADJUSTED TO 4K BOUNDRY.
4762  021642  001006                          BNE     12$             ;BR IF NOT ON 4K BOUNDRY.
4763  021644  046767  157730  157654          BIC     LADMAP, TSTMAP  ;SKIP TESTING LAST BANK.
4764  021652  046767  157724  157650          BIC     LADMAP+2,TSTMAP+2
4765  021660  036767  157702  157712  12$:    BIT     FADMAP, LADMAP  ;CHECK IF FIRST AND LAST IN SAME BANK.
4766  021666  001004                          BNE     13$             ;BR IF IN SAME BANK.
4767  021670  036767  157674  157704          BIT     FADMAP+2,LADMAP+2 ;...UPPER 64K.
4768  021676  001404                          BEQ     14$             ;BR IF FIRST AND LAST NOT SAME BANK.
4769  021700  026767  157670  157654  13$:    CMP     TMPLAD, TMPFAD  ;CHECK IF ANY MEMORY LEFT.
4770  021706  101406                          BLOS    15$             ;BR IF NO MEMORY TO TEST.
4771  021710  005767  157612          14$:    TST     TSTMAP          ;CHECK IF ANY BANKS LEFT TO TEST!!
4772  021714  001017                          BNE     16$             ;BR IF TEST MAP NOT EMPTY.
4773  021716  005767  157606                  TST     TSTMAP+2        ;CHECK FOR ANY BANKS.
4774  021722  001014                          BNE     16$             ;BR IF TEST MAP NOT EMPTY.
4775  021724                          15$:
4776  021724  004567  001310                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
4777  021730  026342                          .WORD   SKPMES          ;ADDRESS OF MESSAGE TO BE TYPED
4778                                                                  ;"SKIPPING TEST #"
4779  021732  005046                          CLR     -(SP)           ;CLEAR THE WORD ON THE STACK.
4780  021734  116716  157142                  MOVB    $TSTNM, (SP)    ;PUT THE DATA ON THE STACK.
4781                          ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOS ROUTINE
4782                          ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
4783  021740  013746  177776                  MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
4784  021744  004767  002436                  JSR     PC,     $TYPOS  ;GO TO THE SUBROUTINE
4785  021750     003                          .BYTE   3               ;TYPE 3 DIGITS.
4786  021751     001                          .BYTE   1               ;TYPE LEADING ZEROS.
4787  021752  000427                          BR      ENDINS          ;RETURN TO SKIP TEST.
4788  021754  062716  000004          16$:    ADD     #4,     (SP)    ;SKIP THE SKIP ON RETURN.
4789  021760  062767  000004  157120          ADD     #4,     $LPADR  ;ADJUST THE LOOP ADR PAST THE SKIP.
4790  021766  012767  017777  157570  20$:    MOV     #MASK4K,FADMSK  ;GET 4K MASK.
4791  021774  016705  157562                  MOV     TMPFAD, R5      ;GET FIRST ADR.
4792  022000  040567  157560          21$:    BIC     R5,     FADMSK  ;CLR MASK ABOVE LOWEST BIT OF FIRST ADR.
4793  022004  006305                          ASL     R5              ;MOVE LOWEST BIT UP ONE.
4794  022006  001374                          BNE     21$             ;LOOP UNTIL OVERFLOW.
4795  022010  012767  017777  157560          MOV     #MASK4K,LADMSK  ;SET MASK BITS
4796  022016  016705  157552                  MOV     TMPLAD, R5      ;GET LAST ADR.
4797  022022  040567  157550          22$:    BIC     R5,     LADMSK  ;CLR ALL MASK BITS ABOVE LOWEST BIT IN LAST ADR.
4798  022026  006305                          ASL     R5              ;MOVE LOWEST BIT OF LAST ADR UP ONE.
4799  022030  001374                          BNE     22$             ;LOOP UNTIL OVERFLOW.
```

```
 4800  022032  000207                  ENDINS: RTS     PC              ;EXIT SCOPE ROUTINE BACK TO TEST.
 4801  022034  000004                  $MXCNT: 4                       ;;MAX. NUMBER OF ITERATIONS
 4802                                  ;*******************************************************************
 4803
 4804                                  .SBTTL  ERROR HANDLER ROUTINE
 4805
 4806                                  ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
 4807                                  ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
 4808                                  ;*AND GO TO $ERRTYP ON ERROR
 4809                                  ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
 4810                                  ;*SW15=1        HALT ON ERROR
 4811                                  ;*SW13=1        INHIBIT ERROR TYPEOUTS
 4812                                  ;*SW10=1        BELL ON ERROR
 4813                                  ;*SW09=1        LOOP ON ERROR
 4814                                  ;*CALL
 4815                                  ;*      ERROR   N       ;;ERROR=EMT AND N=ERROR ITEM NUMBER
 4816
 4817  022036                          $ERROR:
 4818  022036  062716  000002                  ADD     #2,     (SP)    ;ADJUST POINTER PAST CODE WORD.
 4819  022042  105267  157035          7$:     INCB    $ERFLG          ;;SET THE ERROR FLAG
 4820  022046  001775                          BEQ     7$              ;;DON'T LET THE FLAG GO TO ZERO
 4821  022050  016777  157026  157062          MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
 4822  022056  032777  002000  157052          BIT     #BIT10,@SWR     ;;BELL ON ERROR?
 4823  022064  001403                          BEQ     1$              ;;NO - SKIP
 4824  022066  004567  001146                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 4825  022072  001172                          .WORD   $BELL           ;ADDRESS OF MESSAGE TO BE TYPED
 4826  022074  005267  157012          1$:     INC     $ERTTL          ;;COUNT THE NUMBER OF ERRORS
 4827  022100  011667  157012                  MOV     (SP),$ERRPC     ;;GET ADDRESS OF ERROR INSTRUCTION
 4828  022104  162767  000002  157004          SUB     #2,$ERRPC
 4829  022112  117767  157000  156774          MOVB    @$ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
 4830  022120  032777  020000  157010          BIT     #BIT13,@SWR     ;;SKIP TYPEOUT IF SET
 4831  022126  001005                          BNE     20$             ;;SKIP TYPEOUTS
 4832  022130  004767  000074                  JSR     PC,$ERRTYP      ;;GO TO USER ERROR ROUTINE
 4833  022134  004567  001100                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 4834  022140  001177                          .WORD   $CRLF           ;ADDRESS OF MESSAGE TO BE TYPED
 4835  022142                          20$:
 4836  022142  122767  000001  157052          CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
 4837  022150  001007                          BNE     2$              ;;NO,SKIP APT ERROR REPORT
 4838  022152  116767  156736  000004          MOVB    $ITEMB,21$      ;;SET ITEM NUMBER AS ERROR NUMBER
 4839  022160  004767  001400                  JSR     PC,$ATY4        ;;REPORT FATAL ERROR TO APT
 4840  022164     000                  21$:    .BYTE   0
 4841  022165     000                          .BYTE   0
 4842  022166  000777                  22$:    BR      22$             ;;APT ERROR LOOP
 4843  022170  005777  156742          2$:     TST     @SWR            ;;HALT ON ERROR
 4844  022174  100001                          BPL     3$              ;;SKIP IF CONTINUE
 4845  022176  000000                          HALT                    ;;HALT ON ERROR!
 4846  022200  032777  001000  156730  3$:     BIT     #BIT09,@SWR     ;;LOOP ON ERROR SWITCH SET?
 4847  022206  001402                          BEQ     4$              ;;BR IF NO
 4848  022210  016716  156674                  MOV     $LPERR,(SP)     ;;FUDGE RETURN FOR LOOPING
 4849  022214  005767  156750          4$:     TST     $ESCAPE         ;;CHECK FOR AN ESCAPE ADDRESS
 4850  022220  001402                          BEQ     5$              ;;BR IF NONE
 4851  022222  016716  156742                  MOV     $ESCAPE,(SP)    ;;FUDGE RETURN ADDRESS FOR ESCAPE
 4852  022226                          5$:
 4853  022226  000207                          RTS     PC
 4854                                  ;*******************************************************************
 4855
```

```
                                        .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.

022230                                  $ERRTYP:
022230  004567  001004                          JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
022234  001177                                  .WORD   $CRLF           ;ADDRESS OF MESSAGE TO BE TYPED
022236  010046                                  MOV     R0,-(SP)                ;SAVE R0
022240  005000                                  CLR     R0                      ;PICKUP THE ITEM INDEX
022242  156700  156646                          BISB    $ITEMB,R0
022246  001007                                  BNE     1$                      ;IF ITEM NUMBER IS ZERO, JUST
                                                                                ;TYPE THE PC OF THE ERROR
022250  016746  156642                          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**.
022254  013746  177776                          MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
022260  004767  002146                          JSR     PC,     $TYPOC  ;GO TO THE SUBROUTINE
022264  000513                                  BR      10$                     ;GET OUT
022266  016767  156624  157216  1$:             MOV     $ERRPC, $VERPC  ;SET UP VIRTUAL PC FOR TYPEOUT.
022274  166767  156300  157210                  SUB     RELOCF, $VERPC  ;MAKE VIRTUAL IF NOT ALREADY.
022302  005300                                  DEC     R0                      ;ADJUST THE INDEX SO THAT IT WILL
022304  006300                                  ASL     R0                      ;       WORK FOR THE ERROR TABLE
022306  006300                                  ASL     R0
022310  006300                                  ASL     R0
022312  066700  157316                          ADD     .ERRTB, R0      ;FORM TABLE POINTER
022316  012067  000006                          MOV     (R0)+,2$                ;PICKUP "ERROR MESSAGE" POINTER
022322  001406                                  BEQ     3$                      ;SKIP TYPEOUT IF NO POINTER
022324  004567  000710                          JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
022330  000000          2$:             .WORD   0                       ;"ERROR MESSAGE" POINTER GOES HERE
022332  004567  000702                          JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
022336  001177                                  .WORD   $CRLF           ;ADDRESS OF MESSAGE TO BE TYPED
022340  012067  000006          3$:             MOV     (R0)+,4$                ;PICKUP "DATA HEADER" POINTER
022344  001406                                  BEQ     5$                      ;SKIP TYPEOUT IF 0
022346  004567  000666                          JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
022352  000000          4$:             .WORD   0                       ;"DATA HEADER" POINTER GOES HERE
022354  004567  000660                          JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
022360  001177                                  .WORD   $CRLF           ;ADDRESS OF MESSAGE TO BE TYPED
022362  010146          5$:             MOV     R1,-(SP)                ;SAVE R1
022364  012001                                  MOV     (R0)+,R1                ;PICKUP "DATA TABLE" POINTER
022366  001451                                  BEQ     9$                      ;BR IF NO DATA TO BE TYPED
022370  066701  156204                          ADD     RELOCF, R1      ;ADJUST POINTER
022374  012000                                  MOV     (R0)+,R0                ;PICKUP "DATA FORMAT" POINTER
022376  066700  156176                          ADD     RELOCF, R0      ;ADJUST POINTER.
022402  105720          6$:             TSTB    (R0)+                   ;CHECK THE FORMAT
022404  001006                                  BNE     7$                      ;BR IF NOT 16-BIT OCTAL
022406  013146                                  MOV     @(R1)+,-(SP)    ;;SAVE @(R1)+ FOR TYPEOUT
                                        ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOC ROUTINE
                                        ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
022410  013746  177776                          MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
022414  004767  002012                          JSR     PC,     $TYPOC  ;GO TO THE SUBROUTINE
022420  000426                                  BR      8$
022422  100406          7$:             BMI     17$                     ;BRANCH IF NOT DECIMAL
022424  013146                                  MOV     @(R1)+,-(SP)    ;;SAVE @(R1)+ FOR TYPEOUT
```

```
4912                                    ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPDS ROUTINE
4913                                    ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
4914  022426  013746  177776                   MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
4915  022432  004767  001516                   JSR     PC,     $TYPDS  ;GO TO THE SUBROUTINE
4916  022436  000417                           BR      8$              ;SKIP
4917  022440  122760  177777  177777  17$:     CMPB    #-1,    -1(R0)  ;CHECK FOR 18-BIT ADDRESS FORMAT.
4918  022446  001004                           BNE     18$             ;BR IF NOT 18-BIT ADDRESS FORMAT.
4919  022450  013146                           MOV     @(R1)+, -(SP)   ;PUT THE DATA ON THE STACK.
4920  022452  004767  002216                   JSR     PC,     $TYPAD  ;DETERMINE THE PHYSICAL ADDRESS AND TYPE IT.
4921  022456  000407                           BR      8$              ;SKIP
4922  022460                          18$:
4923  022460  005046                           CLR     -(SP)           ;CLEAR THE WORD ON THE STACK.
4924  022462  113116                           MOVB    @(R1)+, (SP)    ;PUT THE DATA ON THE STACK.
4925                                    ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOS ROUTINE
4926                                    ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
4927  022464  013746  177776                   MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
4928  022470  004767  001712                   JSR     PC,     $TYPOS  ;GO TO THE SUBROUTINE
4929  022474     003                           .BYTE   3               ;TYPE 3 DIGITS.
4930  022475     001                           .BYTE   1               ;TYPE LEADING ZEROS.
4931  022476  005711                  8$:      TST     (R1)                    ;IS THERE ANOTHER NUMBER?
4932  022500  001404                           BEQ     9$                      ;BR IF NO
4933  022502  004567  000532                   JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
4934  022506  022526                           .WORD   11$             ;ADDRESS OF MESSAGE TO BE TYPED
4935  022510  000734                           BR      6$                      ;LOOP
4936
4937  022512  012601                  9$:      MOV     (SP)+,R1                ;RESTORE R1
4938  022514  012600                  10$:     MOV     (SP)+,R0                ;RESTORE R0
4939  022516  004567  000516                   JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
4940  022522  001177                           .WORD   $CRLF           ;ADDRESS OF MESSAGE TO BE TYPED
4941  022524  000207                           RTS     PC                      ;RETURN
4942  022526  000011                  11$:     .ASCIZ  /       /       ;TAB CHARACTER.
4943                                           .EVEN
4944                                    ;******************************************************************
4945
4946                                    .SBTTL  TTY INPUT ROUTINE
4947
4948                                    ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
4949                                    ;*CALL:
4950                                    ;*      RDCHR                   ;;INPUT A SINGLE CHARACTER FROM THE TTY
4951                                    ;*      RETURN HERE             ;;CHARACTER IS ON THE STACK
4952                                    ;
4953
4954  022530  011646                  $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC
4955  022532  016666  000004  000002          MOV     4(SP),2(SP)     ;;SAVE THE PS
4956  022540  105777  156376          1$:     TSTB    @$TKS           ;;WAIT FOR
4957  022544  100375                          BPL     1$              ;;A CHARACTER
4958  022546  117766  156372  000004          MOVB    @$TKB,4(SP)     ;;READ THE TTY
4959  022554  042766  177600  000004          BIC     #^C<177>,4(SP)  ;;GET RID OF JUNK IF ANY
4960  022562  000002                          RTI                     ;;GO BACK TO USER
4961                                    ;******************************************************************
4962                                    ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
4963                                    ;*CALL:
4964                                    ;*      RDLIN                   ;;INPUT A STRING FROM THE TTY
4965                                    ;*      RETURN HERE             ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
4966                                    ;*                              ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
4967
```

```
 4968  022564  010346                  $RDLIN: MOV     R3,-(SP)        ;;SAVE R3
 4969  022566  005046                          CLR     -(SP)           ;;CLEAR THE RUBOUT KEY
 4970  022570  013703  023055            1$:   MOV     @#$TTYIN,R3     ;;GET ADDRESS
 4971  022574  022703  023065            2$:   CMP     @#$TTYIN+8.,R3  ;;BUFFER FULL?
 4972  022600  101467                          BLOS    4$              ;;BR IF YES
 4973                                    ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $RDCHR ROUTINE
 4974                                    ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
 4975  022602  013746  177776                    MOV     @#PSW, -(SP)    ;PUT THE PROCESSOR STATUS ON THE STACK
 4976  022606  004767  177716                    JSR     PC,    $RDCHR   ;GO TO THE SUBROUTINE
 4977  022612  112613                          MOVB    (SP)+,(R3)      ;;GET CHARACTER
 4978  022614  122713  000177                  CMPB    #177,(R3)       ;;IS IT A RUBOUT
 4979  022620  001024                          BNE     5$              ;;BR IF NO
 4980  022622  005716                          TST     (SP)            ;;IS THIS THE FIRST RUBOUT?
 4981  022624  001010                          BNE     6$              ;;BR IF NO
 4982  022626  112767  000134  000212        MOVB    #'\,9$          ;;TYPE A BACK SLASH
 4983  022634  004567  000400                  JSR     R5,    $PRINT   ;GO PRINT OUT THE FOLLOWING MESSAGE.
 4984  022640  023046                          .WORD   9$              ;ADDRESS OF MESSAGE TO BE TYPED
 4985  022642  012716  177777                  MOV     #-1,(SP)        ;;SET THE RUBOUT KEY
 4986  022646  005303                  6$:     DEC     R3              ;;BACKUP BY ONE
 4987  022650  020327  023055                  CMP     R3,@#$TTYIN     ;;STACK EMPTY?
 4988  022654  103441                          BLO     4$              ;;BR IF YES
 4989  022656  111367  000164                  MOVB    (R3),9$         ;;SETUP TO TYPEOUT THE DELETED CHAR.
 4990  022662  004567  000352                  JSR     R5,    $PRINT   ;GO PRINT OUT THE FOLLOWING MESSAGE.
 4991  022666  023046                          .WORD   9$              ;ADDRESS OF MESSAGE TO BE TYPED
 4992  022670  000741                          BR      2$              ;;GO READ ANOTHER CHAR.
 4993  022672  005716                  5$:     TST     (SP)            ;;RUBOUT KEY SET?
 4994  022674  001407                          BEQ     7$              ;;BR IF NO
 4995  022676  112767  000134  000142        MOVB    #'\,9$          ;;TYPE A BACK SLASH
 4996  022704  004567  000330                  JSR     R5,    $PRINT   ;GO PRINT OUT THE FOLLOWING MESSAGE.
 4997  022710  023046                          .WORD   9$              ;ADDRESS OF MESSAGE TO BE TYPED
 4998  022712  005016                          CLR     (SP)            ;;CLEAR THE RUBOUT KEY
 4999  022714  122713  000025          7$:     CMPB    #25,(R3)        ;;IS CHARACTER A CTRL U?
 5000  022720  001004                          BNE     8$              ;;BR IF NO
 5001  022722  004567  000312                  JSR     R5,    $PRINT   ;GO PRINT OUT THE FOLLOWING MESSAGE.
 5002  022726  023050                          .WORD   $CNTLU          ;ADDRESS OF MESSAGE TO BE TYPED
 5003  022730  000717                          BR      1$              ;;GO START OVER
 5004  022732  122713  000012          8$:     CMPB    #12,(R3)        ;;IS CHARACTER A "LF"?
 5005  022736  001014                          BNE     3$              ;;BRANCH IF NO
 5006  022740  105013                          CLRB    (R3)            ;;CLEAR THE CHARACTER
 5007  022742  004567  000272                  JSR     R5,    $PRINT   ;GO PRINT OUT THE FOLLOWING MESSAGE.
 5008  022746  001177                          .WORD   $CRLF           ;ADDRESS OF MESSAGE TO BE TYPED
 5009  022750  004567  000264                  JSR     R5,    $PRINT   ;GO PRINT OUT THE FOLLOWING MESSAGE.
 5010  022754  023055                          .WORD   $TTYIN          ;ADDRESS OF MESSAGE TO BE TYPED
 5011  022756  000706                          BR      2$              ;;GO PICKUP ANOTHER CHACTER
 5012  022760                          4$:
 5013  022760  004567  000254                  JSR     R5,    $PRINT   ;GO PRINT OUT THE FOLLOWING MESSAGE.
 5014  022764  001176                          .WORD   $QUES           ;ADDRESS OF MESSAGE TO BE TYPED
 5015  022766  000700                          BR      1$              ;;CLEAR THE BUFFER AND LOOP
 5016  022770  111367  000052          3$:     MOVB    (R3),9$         ;;ECHO THE CHARACTER
 5017  022774  004567  000240                  JSR     R5,    $PRINT   ;GO PRINT OUT THE FOLLOWING MESSAGE.
 5018  023000  023046                          .WORD   9$              ;ADDRESS OF MESSAGE TO BE TYPED
 5019  023002  122723  000015                  CMPB    #15,(R3)+       ;;CHECK FOR RETURN
 5020  023006  001272                          BNE     2$              ;;LOOP IF NOT RETURN
 5021  023010  105063  177777                  CLRB    -1(R3)          ;;CLEAR RETURN (THE 15)
 5022  023014  004567  000220                  JSR     R5,    $PRINT   ;GO PRINT OUT THE FOLLOWING MESSAGE.
 5023  023020  001200                          .WORD   $LF             ;ADDRESS OF MESSAGE TO BE TYPED
```

```
5024  023022  005726                          TST     (SP)+           ;;CLEAN RUBOUT KEY FROM THE STACK
5025  023024  012603                          MOV     (SP)+,R3        ;;RESTORE R3
5026  023026  011646                          MOV     (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
5027  023030  016666  000004  000002          MOV     4(SP),2(SP)     ;;      FIRST ASCII CHARACTER ON IT
5028  023036  012766  023055  000004          MOV     #$TTYIN,4(SP)
5029  023044  000002                          RTI                     ;;RETURN
5030  023046     000                  9$:     .BYTE   0               ;;STORAGE FOR ASCII CHAR. TO TYPE
5031  023047     000                          .BYTE   0               ;;TERMINATOR
5032  023050  052536  005015     000  $CNTLU: .ASCIZ  /↑U/<15><12>    ;;CONTROL "U"
5033  023055  000010                  $TTYIN: .BLKB   8.              ;;RESERVE 8 BYTES FOR TTY INPUT
5034          023066                  .EVEN
5035                                  ;*****************************************************************
5036
5037                                  .SBTTL  READ AN OCTAL NUMBER FROM THE TTY
5038
5039                                  ;*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
5040                                  ;*CHANGE IT TO BINARY.
5041                                  ;*THE INPUT CHARACTERS WILL BE CHECKED TO INSURED THEY ARE LEGAL
5042                                  ;*OCTAL DIGITS. IF AN ILLEGAL CHARACTER IS READ A "?" WILL BE TYPED
5043                                  ;*FOLLOWED BY A CARRIAGE RETURN-LINE FEED. THE COMPLETE NUMBER MUST
5044                                  ;*THEN BE RETYPED. THE INPUT IS TERMINATED BY TYPING A CARRIAGE RETURN.
5045                                  ;*CALL:
5046                                  ;*      RDOCT                   ;;READ AN OCTAL NUMBER
5047                                  ;*      RETURN HERE             ;;LOW ORDER BITS ARE ON TOP OF THE STACK
5048                                  ;*                              ;;HIGH ORDER BITS ARE IN $HIOCT
5049
5050  023066  011646                  $RDOCT: MOV     (SP),-(SP)      ;;PROVIDE SPACE FOR THE
5051  023070  016666  000004  000002          MOV     4(SP),2(SP)     ;;INPUT NUMBER
5052  023076  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
5053  023100  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
5054  023102  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
5055  023104                          1$:
5056                                  ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $RDLIN ROUTINE
5057                                  ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
5058  023104  013746  177776                  MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
5059  023110  004767  177450                  JSR     PC,     $RDLIN  ;GO TO THE SUBROUTINE
5060  023114  012600                          MOV     (SP)+,R0        ;;GET ADDRESS OF 1ST CHARACTER
5061  023116  010067  000102                  MOV     R0,5$           ;;AND SAVE IT
5062  023122  005001                          CLR     R1              ;;CLEAR DATA WORD
5063  023124  005002                          CLR     R2
5064  023126  112046                  2$:     MOVB    (R0)+,-(SP)     ;;PICKUP THIS CHARACTER
5065  023130  001420                          BEQ     3$              ;;IF ZERO GET OUT
5066  023132  122716  000060                  CMPB    #'0,(SP)        ;;MAKE SURE THIS CHARACTER
5067  023136  003026                          BGT     4$              ;;IS AN OCTAL DIGIT
5068  023140  122716  000067                  CMPB    #'7,(SP)
5069  023144  002423                          BLT     4$
5070  023146  006301                          ASL     R1              ;;*2
5071  023150  006102                          ROL     R2
5072  023152  006301                          ASL     R1              ;;*4
5073  023154  006102                          ROL     R2
5074  023156  006301                          ASL     R1              ;;*8
5075  023160  006102                          ROL     R2
5076  023162  042716  177770                  BIC     #↑C7,(SP)       ;;STRIP THE ASCII JUNK
5077  023166  062601                          ADD     (SP)+,R1        ;;ADD IN THIS DIGIT
5078  023170  000756                          BR      2$              ;;LOOP
5079  023172  005726                  3$:     TST     (SP)+           ;;CLEAN TERMINATOR FROM STACK
```

```
5080  023174  010166  000012                  MOV     R1,12(SP)       ;;SAVE THE RESULT
5081  023200  010267  000032                  MOV     R2,$HIOCT
5082  023204  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
5083  023206  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
5084  023210  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
5085  023212  000002                          RTI                     ;;RETURN
5086  023214  005726                  4$:     TST     (SP)+           ;;CLEAN PARTIAL FROM STACK
5087  023216  105010                          CLRB    (R0)            ;;SET A TERMINATOR
5088  023220  004567  000014                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
5089  023224  000000                  5$:     .WORD   0
5090  023226  004567  000006                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
5091  023232  001176                          .WORD   $QUES           ;ADDRESS OF MESSAGE TO BE TYPED
5092  023234  000723                          BR      1$              ;;TRY AGAIN
5093  023236  000000                  $HIOCT: .WORD   0               ;;HIGH ORDER BITS GO HERE
5094
5095                                  ;*****************************************************************
5096                                  ;* SUBROUTINE TO PASS RELOCATED MESSAGE ADDRESSES TO THE $TYPE ROUTINE.
5097                                  ;*  CALL:       JSR     R5,     $PRINT
5098                                  ;*              <MESSAGE VIRTUAL ADDRESS>
5099                                  ;*****************************************************************
5100  023240  012567  000016          $PRINT: MOV     (R5)+,  1$      ;GET THE MESSAGE VIRTUAL ADDRESS.
5101  023244  066767  155330  000010          ADD     RELOCF, 1$      ;MAKE IT PHYSICAL.
5102                                  ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPE ROUTINE
5103                                  ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
5104  023252  013746  177776                  MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
5105  023256  004767  000004                  JSR     PC,     $TYPE   ;GO TO THE SUBROUTINE
5106  023262  000000                  1$:     .WORD   0               ;CONTAINS THE PHYSICAL MESSAGE ADDRESS.
5107  023264  000205                          RTS     R5              ;RETURN.
5108
5109                                  ;*****************************************************************
5110
5111                                  .SBTTL  TYPE ROUTINE
5112
5113                                  ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
5114                                  ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
5115                                  ;*NOTE1:        $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
5116                                  ;*NOTE2:        $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
5117                                  ;*NOTE3:        $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
5118                                  ;*
5119                                  ;*CALL:
5120                                  ;*1) USING A TRAP INSTRUCTION
5121                                  ;*      TYPE    ,MESADR         ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
5122                                  ;*OR
5123                                  ;*      TYPE
5124                                  ;*      MESADR
5125                                  ;*
5126
5127  023266  105767  155663          $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
5128  023272  100002                          BPL     1$              ;;BR IF YES
5129  023274  000000                          HALT                    ;;HALT HERE IF NO TERMINAL
5130  023276  000430                          BR      3$              ;;LEAVE
5131  023300  010046                  1$:     MOV     R0,-(SP)        ;;SAVE R0
5132  023302  017600  000002                  MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
5133  023306  122767  000001  155706          CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
5134  023314  001011                          BNE     62$             ;;NO,GO CHECK FOR APT CONSOLE
5135  023316  132767  000100  155677          BITB    #APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
```

```
5136  023324  001405                          BEQ     62$             ;;NO,GO CHECK FOR CONSOLE
5137  023326  010067  000004                  MOV     R0,61$          ;;SETUP MESSAGE ADDRESS FOR APT
5138  023332  004767  000216                  JSR     PC,$ATY3        ;;SPOOL MESSAGE TO APT
5139  023336  000000                  61$:    .WORD   0               ;;MESSAGE ADDRESS
5140  023340  132767  000040  155655  62$:    BITB    #APTCSUP,$ENVM  ;;APT CONSOLE SUPPRESSED
5141  023346  001003                          BNE     60$             ;;YES,SKIP TYPE OUT
5142  023350  112046                  2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
5143  023352  001005                          BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
5144  023354  005726                          TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
5145  023356  012600                  60$:    MOV     (SP)+,R0        ;;RESTORE R0
5146  023360  062716  000002          3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
5147  023364  000002                          RTI                     ;;RETURN
5148  023366  122716  000011          4$:     CMPB    #THT,(SP)       ;;BRANCH IF <HT>
5149  023372  001427                          BEQ     8$
5150  023374  122716  000200                  CMPB    #TCRLF,(SP)     ;;BRANCH IF NOT <CRLF>
5151  023400  001005                          BNE     5$
5152  023402  005726                          TST     (SP)+           ;;POP  <CR><LF> EQUIV
5153  023404  004567  177630                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
5154  023410  001177                          $CRLF
5155  023412  000756                          BR      2$              ;;GET NEXT CHARACTER
5156  023414  004767  000056          5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
5157  023420  126726  155530          6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
5158  023424  001351                          BNE     2$              ;;IF NO GO GET NEXT CHAR.
5159  023426  016746  155520                  MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
5160                                                                  ;;AND THE NULL CHAR.
5161  023432  105366  000001          7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
5162  023436  002770                          BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
5163  023440  004767  000032                  JSR     PC,$TYPEC       ;;GO TYPE A NULL
5164  023444  105367  000072                  DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
5165  023450  000770                          BR      7$              ;;LOOP
5166
5167                                  ;HORIZONTAL TAB PROCESSOR
5168
5169  023452  112716  000040          8$:     MOVB    #40,(SP)        ;;REPLACE TAB WITH SPACE
5170  023456  004767  000014          9$:     JSR     PC,$TYPEC       ;;TYPE A SPACE
5171  023462  132767  000007  000052          BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
5172  023470  001372                          BNE     9$              ;;TAB STOP
5173  023472  005726                          TST     (SP)+           ;;POP SPACE OFF STACK
5174  023474  000725                          BR      2$              ;;GET NEXT CHARACTER
5175  023476  105777  155444          $TYPEC: TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
5176  023502  100375                          BPL     $TYPEC
5177  023504  116677  000002  155436          MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
5178  023512  122766  000015  000002          CMPB    #15,2(SP)       ;;BRANCH IF
5179  023520  001003                          BNE     1$              ;;NOT <CR>
5180  023522  105067  000014                  CLRB    $CHARCNT        ;;
5181  023526  000406                          BR      $TYPEX          ;;EXIT
5182  023530  122766  000012  000002  1$:     CMPB    #12,2(SP)       ;;BRANCH IF
5183  023536  002002                          BGE     $TYPEX          ;;<LF>
5184  023540  105227                          INCB    (PC)+           ;;INC SPACE
5185  023542  000000                  $CHARCNT:.WORD  0               ;;COUNT
5186  023544  000207                  $TYPEX: RTS     PC
5187                                  ;;      EQUATES
5188          000011                  THT=11
5189          000200                  TCRLF=200
5190
5191                                  ;*****************************************************************
```

```
5192
5193                                            .SBTTL  APT COMMUNICATIONS ROUTINE
5194  023546  112767  000001  000376  $ATY1:    MOVB    #1,$FFLG        ;TO REPORT FATAL ERROR
5195  023554  112767  000001  000366  $ATY3:    MOVB    #1,$MFLG        ;TO TYPE A MESSAGE
5196  023562  000403                            BR      $ATYC
5197  023564  112767  000001  000360  $ATY4:    MOVB    #1,$FFLG        ;TO ONLY REPORT FATAL ERROR
5198  023572                          $ATYC:
5199  023572  010046                            MOV     R0,-(SP)        ;;PUSH R0 ON STACK
5200  023574  010146                            MOV     R1,-(SP)        ;;PUSH R1 ON STACK
5201  023576  105767  000346                    TSTB    $MFLG           ;SHOULD TYPE A MESSAGE?
5202  023602  001450                            BEQ     5$              ;IF NOT:  BR
5203  023604  122767  000001  155410            CMPB    #APTENV,$ENV    ;OPERATING UNDER APT?
5204  023612  001031                            BNE     3$              ;IF NOT:  BR
5205  023614  132767  000100  155401            BITB    #APTSPOOL,$ENVM ;SHOULD SPOOL MESSAGES?
5206  023622  001425                            BEQ     3$              ;IF NOT:  BR
5207  023624  017600  000004                    MOV     @4(SP),R0       ;GET MESSAGE ADDR.
5208  023630  062766  000002  000004            ADD     #2,4(SP)                ;BUMP RETURN ADDR.
5209  023636  005767  155340          1$:       TST     $MSGTYPE        ;SEE IF DONE W/ LAST XMISSION?
5210  023642  001375                            BNE     1$              ;IF NOT:  WAIT
5211  023644  010067  155346                    MOV     R0,$MSGAD       ;PUT ADDR IN MAILBOX
5212  023650  105720                  2$:       TSTB    (R0)+           ;FIND END OF MESSAGE
5213  023652  001376                            BNE     2$
5214  023654  166700  155336                    SUB     $MSGAD,R0       ;SUB START OF MESSAGE
5215  023660  006200                            ASR     R0              ;GET MESSAGE LNGTH IN WORDS
5216  023662  010067  155332                    MOV     R0,$MSGLGT      ;PUT LENGTH IN MAILBOX
5217  023666  012767  000004  155306            MOV     #4,$MSGTYPE     ;TELL APT TO TAKE MSG.
5218  023674  000413                            BR      5$
5219  023676  017667  000004  000016  3$:       MOV     @4(SP),4$       ;PUT MSG ADDR IN JSR LINKAGE
5220  023704  062766  000002  000004            ADD     #2,4(SP)                ;BUMP RETURN ADDRESS
5221  023712  016746  154060                    MOV     177776,-(SP)    ;;PUSH 177776 ON STACK
5222  023716  004767  177344                    JSR     PC,$TYPE        ;CALL TYPE MACRO
5223  023722  000000                  4$:       .WORD   0
5224  023724                          5$:
5225  023724  105767  000221                    TSTB    $LFLG           ;SHOULD LOG AN ERROR?
5226  023730  001422                            BEQ     10$             ;IF NOT:  BR
5227  023732  017600  000004                    MOV     @4(SP),R0               ;GET ERROR #
5228  023736  062766  000002  000004            ADD     #2,4(SP)                ;BUMP RETURN ADDR.
5229  023744  012701  001342                    MOV     #$ASTAT,R1      ;POINT TO TABLE START
5230  023750  005711                  6$:       TST     (R1)            ;END OF TABLE?
5231  023752  100404                            BMI     8$              ;IF SO:  BR
5232  023754  020021                            CMP     R0,(R1)+        ;PROPER ENTRY?
5233  023756  001406                            BEQ     9$              ;IF SO:  BR
5234  023760  005721                            TST     (R1)+           ;MOVE PAST COUNTER WORD
5235  023762  000772                            BR      6$              ;KEEP LOOKING
5236  023764  026701  155520          8$:       CMP     $APTR,R1        ;TABLE FULL?
5237  023770  001402                            BEQ     10$             ;IF SO:  BR -- NO MORE ROOM
5238  023772  010021                            MOV     R0,(R1)+        ;SET UP NEW ENTRY
5239  023774  005211                  9$:       INC     (R1)            ;BUMP ERROR COUNT
5240  023776  105767  000150          10$:      TSTB    $FFLG           ;SHOULD REPORT FATAL ERROR?
5241  024002  001416                            BEQ     12$             ;IF NOT:  BR
5242  024004  005767  155212                    TST     $ENV            ;RUNNING UNDER APT?
5243  024010  001413                            BEQ     12$             ;IF NOT:  BR
5244  024012  005767  155164          11$:      TST     $MSGTYPE        ;FINISHED LAST MESSAGE?
5245  024016  001375                            BNE     11$             ;IF NOT:  WAIT
5246  024020  017667  000004  155156            MOV     @4(SP),$FATAL   ;GET ERROR #
5247  024026  062766  000002  000004            ADD     #2,4(SP)                ;BUMP RETURN ADDR.
```

```
5248  024034  005267  155142          INC     $MSGTYPE        ;TELL APT TO TAKE ERROR
5249  024040  105067  000106  12$:    CLRB    $FFLG           ;CLEAR FATAL FLAG
5250  024044  105067  000101          CLRB    $LFLG           ;CLEAR LOG FLAG
5251  024050  105067  000074          CLRB    $MFLG           ;CLEAR MESSAGE FLAG
5252  024054  012601                  MOV     (SP)+,R1        ;;POP STACK INTO R1
5253  024056  012600                  MOV     (SP)+,R0        ;;POP STACK INTO R0
5254  024060  000207                  RTS     PC              ;RETURN
5255
5256  024062                  $ATY6:
5257  024062  010046                  MOV     R0,-(SP)        ;;PUSH R0 ON STACK
5258  024064  016700  155420          MOV     $APTR,R0
5259  024070  162700  001342          SUB     #$ASTAT,R0      ;GET SIZE OF STAT TABLE
5260  024074  005767  155102  1$:     TST     $MSGTY          ;SEE IF DONE LAST COMMUNICATION
5261  024100  001375                  BNE     1$              ;IF NOT:  WAIT
5262  024102  010067  155112          MOV     R0,$MSGLG       ;SET MESSAGE LENGTH
5263  024106  012767  001342  155102  MOV     #$ASTAT,$MSGAD  ;SET MESSAGE ADDR.
5264  024114  012767  000002  155060  MOV     #2,$MSGTY       ;TELL APT TO TAKE STATS.
5265  024122  012600                  MOV     (SP)+,R0        ;;POP STACK INTO R0
5266  024124  000207                  RTS     PC              ;RETURN
5267  024126                  $ATY7:
5268  024126  010046                  MOV     R0,-(SP)        ;;PUSH R0 ON STACK
5269  024130  012701  001342          MOV     #$ASTAT,R1      ;GET START OF TABLE
5270  024134  005721          1$:     TST     (R1)+           ;END OF TABLE?
5271  024136  100402                  BMI     2$              ;IF SO:  BR
5272  024140  005021                  CLR     (R1)+           ;CLEAR ERROR COUNT
5273  024142  000774                  BR      1$              ;KEEP CLEARING
5274  024144                  2$:
5275  024144  012600                  MOV     (SP)+,R0        ;;POP STACK INTO R0
5276  024146  000207                  RTS     PC              ;RETURN
5277  024150     000          $MFLG:  .BYTE   0               ;MESSG. FLAG
5278  024151     000          $LFLG:  .BYTE   0               ;LOG FLAG
5279  024152     000          $FFLG:  .BYTE   0               ;FATAL FLAG
5280          024154                  .EVEN
5281          000200          APTSIZE=200
5282          000001          APTENV=001
5283          000100          APTSPOOL=100
5284          000040          APTCSUP=040
5285                          ;*****************************************************************
5286
5287                          .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
5288
5289                          ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
5290                          ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
5291                          ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
5292                          ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
5293                          ;*REPLACED WITH SPACES.
5294                          ;*CALL:
5295                          ;*      MOV     NUM,-(SP)       ;;PUT THE BINARY NUMBER ON THE STACK
5296                          ;*      TYPDS                   ;;GO TO THE ROUTINE
5297
5298  024154                  $TYPDS:
5299  024154  010046                  MOV     R0,-(SP)        ;;PUSH R0 ON STACK
5300  024156  010146                  MOV     R1,-(SP)        ;;PUSH R1 ON STACK
5301  024160  010246                  MOV     R2,-(SP)        ;;PUSH R2 ON STACK
5302  024162  010346                  MOV     R3,-(SP)        ;;PUSH R3 ON STACK
5303  024164  010546                  MOV     R5,-(SP)        ;;PUSH R5 ON STACK
```

```
5304  024166  012746  020200                        MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
5305  024172  016605  000020                        MOV     20(SP),R5       ;;GET THE INPUT NUMBER
5306  024176  100004                                BPL     1$              ;;BR IF INPUT IS POS.
5307  024200  005405                                NEG     R5              ;;MAKE THE BINARY NUMBER POS.
5308  024202  112766  000055  000001                MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
5309  024210  016700  154364              1$:       MOV     RELOCF, R0      ;GET RELOCATION FACTOR.
5310  024214  012703  024376                        MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
5311  024220  060003                                ADD     R0,     R3      ;ADD IN RELOCATION FACTOR.
5312  024222  112723  000040                        MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
5313  024226  005002                        2$:     CLR     R2              ;;CLEAR THE BCD NUMBER
5314  024230  016001  024366                        MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
5315  024234  160105                        3$:     SUB     R1,R5           ;;FORM THIS BCD DIGIT
5316  024236  002402                                BLT     4$              ;;BR IF DONE
5317  024240  005202                                INC     R2              ;;INCREASE THE BCD DIGIT BY 1
5318  024242  000774                                BR      3$
5319  024244  060105                        4$:     ADD     R1,R5           ;;ADD BACK THE CONSTANT
5320  024246  005702                                TST     R2              ;;CHECK IF BCD DIGIT=0
5321  024250  001002                                BNE     5$              ;;FALL THROUGH IF 0
5322  024252  105716                                TSTB    (SP)            ;;STILL DOING LEADING 0'S?
5323  024254  100407                                BMI     7$              ;;BR IF YES
5324  024256  106316                        5$:     ASLB    (SP)            ;;MSD?
5325  024260  103003                                BCC     6$              ;;BR IF NO
5326  024262  116663  000001  177777                MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
5327  024270  052702  000060                6$:     BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
5328  024274  052702  000040                7$:     BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
5329  024300  110223                                MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
5330  024302  005720                                TST     (R0)+           ;;JUST INCREMENTING
5331  024304  020067  155326                        CMP     R0,     .EIGHT  ;CHECK THE TABLE INDEX
5332  024310  103746                                BLO     2$              ;GO DO THE NEXT DIGIT
5333  024312  101002                                BHI     8$              ;;GO TO EXIT
5334  024314  010502                                MOV     R5,R2           ;;GET THE LSD
5335  024316  000764                                BR      6$              ;;GO CHANGE TO ASCII
5336  024320  105726                        8$:     TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
5337  024322  100003                                BPL     9$              ;;BR IF NO
5338  024324  116663  177777  177776                MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
5339  024332  105013                        9$:     CLRB    (R3)            ;;SET THE TERMINATOR
5340  024334  012605                                MOV     (SP)+,R5        ;;POP STACK INTO R5
5341  024336  012603                                MOV     (SP)+,R3        ;;POP STACK INTO R3
5342  024340  012602                                MOV     (SP)+,R2        ;;POP STACK INTO R2
5343  024342  012601                                MOV     (SP)+,R1        ;;POP STACK INTO R1
5344  024344  012600                                MOV     (SP)+,R0        ;;POP STACK INTO R0
5345  024346  004567  176666                        JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
5346  024352  024376                                .WORD   $DBLK           ;ADDRESS OF MESSAGE TO BE TYPED
5347  024354  016666  000002  000004                MOV     2(SP),4(SP)     ;;ADJUST THE STACK
5348  024362  012616                                MOV     (SP)+,(SP)
5349  024364  000002                                RTI                     ;;RETURN TO USER
5350  024366  023420                        $DTBL:  10000.
5351  024370  001750                                1000.
5352  024372  000144                                100.
5353  024374  000012                                10.
5354  024376  000004                        $DBLK:  .BLKW   4
5355                                        ;*********************************************************************
5356
5357                                        .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
5358
5359                                        ;*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

024406  017646  000000          $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
024412  116667  000001  000213          MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
024420  112667  000211                  MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
024424  062716  000002                  ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
024430  000406                          BR      $TYPON
024432  112767  000001  000173  $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
024440  112767  000006  000167          MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
024446  112767  000005  000156  $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
024454  010346                          MOV     R3,-(SP)        ;;SAVE R3
024456  010446                          MOV     R4,-(SP)        ;;SAVE R4
024460  010546                          MOV     R5,-(SP)        ;;SAVE R5
024462  116704  000147                  MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
024466  005404                          NEG     R4
024470  062704  000006                  ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
024474  110467  000134                  MOVB    R4,$OMODE       ;;SAVE IT FOR USE
024500  116704  000127                  MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
024504  016605  000012                  MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
024510  005003                          CLR     R3              ;;CLEAR THE OUTPUT WORD
024512  006105                  1$:     ROL     R5              ;;ROTATE MSB INTO "C"
024514  000404                          BR      3$              ;;GO DO MSB
024516  006105                  2$:     ROL     R5              ;;FORM THIS DIGIT
024520  006105                          ROL     R5
024522  006105                          ROL     R5
024524  010503                          MOV     R5,R3
024526  006103                  3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
024530  105367  000100                  DECB    $OMODE          ;;TYPE THIS DIGIT?
024534  100017                          BPL     7$              ;;BR IF NO
024536  042703  177770                  BIC     #177770,R3      ;;GET RID OF JUNK
024542  001002                          BNE     4$              ;;TEST FOR 0
024544  005704                          TST     R4              ;;SUPPRESS THIS 0?
024546  001403                          BEQ     5$              ;;BR IF YES
024550  005204                  4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
024552  052703  000060                  BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
024556  052703  000040          5$:     BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
024562  110367  000042                  MOVB    R3,8$           ;;SAVE FOR TYPING
```

```
5416  024566  004567  176446                JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
5417  024572  024630                        .WORD   8$              ;ADDRESS OF MESSAGE TO BE TYPED
5418  024574  105367  000032        7$:     DECB    $OCNT           ;;COUNT BY 1
5419  024600  003346                        BGT     2$              ;;BR IF MORE TO DO
5420  024602  002402                        BLT     6$              ;;BR IF DONE
5421  024604  005204                        INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
5422  024606  000743                        BR      2$              ;;GO DO THE LAST DIGIT
5423  024610  012605                6$:     MOV     (SP)+,R5        ;;RESTORE R5
5424  024612  012604                        MOV     (SP)+,R4        ;;RESTORE R4
5425  024614  012603                        MOV     (SP)+,R3        ;;RESTORE R3
5426  024616  016666  000002  000004        MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
5427  024624  012616                        MOV     (SP)+,(SP)
5428  024626  000002                        RTI                     ;;RETURN
5429  024630     000                8$:     .BYTE   0               ;;STORAGE FOR ASCII DIGIT
5430  024631     000                        .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
5431  024632     000        $OCNT:  .BYTE   0               ;;OCTAL DIGIT COUNTER
5432  024633     000        $OFILL: .BYTE   0               ;;ZERO FILL SWITCH
5433  024634  000000        $OMODE: .WORD   0               ;;NUMBER OF DIGITS TO TYPE
5434                        ;ERROR TRAP SERVICE ROUTINE
5435  024636  005727        ERRTRP: TST     (PC)+           ;CHECK IF PREV TRAP TO 4 REPORTED
5436  024640  000000        1$:     .WORD   0               ;CONTAINS ERROR REPORTED FLAG
5437  024642  001010                BNE     2$              ;BRANCH IF NOT REPORTED
5438  024644  005267  177770        INC     1$              ;SET DOUBLE TRAP FLAG.
5439  024650  011667  154310        MOV     (SP),   $TMP3   ;SAVE THE BAD PC FOR TYPOUT.
5440  024654  004767  175156        JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
5441  024660  000031                .WORD   31              ;ERROR TYPE CODE.
5442  024662  000401                BR      3$              ;SKIP HALT
5443  024664  000000        2$:     HALT                    ;ERROR! SECOND TRAP TO 4 OCCURRED
5444                                                        ;BEFORE FIRST WAS PRINTED
5445  024666  005067  177746  3$:     CLR     1$
5446  024672  000002                RTI                     ;RETURN TO PROGRAM AND TRY TO RECOVER
5447
5448                        .SBTTL  PHYSICAL ADDRESS TYPE ROUTINE
5449                        ;*  ROUTINE TO TYPE A PHYSICAL ADDRESS (18 BITS).
5450  024674        $TYPAD:
5451  024674  010046                MOV     R0,-(SP)        ;;PUSH R0 ON STACK
5452  024676  010146                MOV     R1,-(SP)        ;;PUSH R1 ON STACK
5453  024700  010246                MOV     R2,-(SP)        ;;PUSH R2 ON STACK
5454  024702  010346                MOV     R3,-(SP)        ;;PUSH R3 ON STACK
5455  024704  016602  000012        MOV     12(SP), R2      ;GET BASE ADDRESS
5456  024710  005003                CLR     R3              ;WORKING & INDEX REGISTER
5457  024712  005767  153670        TST     MMAVA           ;CHECK FOR MEM MGMT AVAILABLE
5458  024716  001430                BEQ     1$              ;BRANCH IF NO MEM MGMT
5459  024720  032737  000001  177572        BIT     #1,     @#SR0   ;CHECK IF MEM MGMT ENABLED
5460  024726  001424                BEQ     1$              ;BRANCH IF MEM MGMT NOT ENABLED
5461  024730  010201                MOV     R2,     R1      ;COPY VIRTUAL ADR
5462  024732  006101                ROL     R1              ;SHUFFLE BITS 13,14,15 INTO 1,2,3
5463  024734  006101                ROL     R1
5464  024736  006101                ROL     R1
5465  024740  006101                ROL     R1
5466  024742  006101                ROL     R1
5467  024744  042701  177761        BIC     #177761, R1     ;CLR ALL EXCEPT BITS 1,2,3
5468  024750  062701  172340        ADD     #KIPAR0, R1     ;SET TO APPROPRIATE PAR
5469  024754  011101                MOV     (R1),   R1      ;GET CONTENTS OF PAR
5470  024756  012700  000006        MOV     #6,     R0      ;SET UP COUNTER
5471  024762  006301        4$:     ASL     R1              ;SHIFT PAR
```

```
5472  024764  006103                        ROL     R3              ;SAVE OVERFLOW BITS
5473  024766  077003                        SOB     R0,     4$      ;COUNT SIX SHIFTS
5474  024770  042702  160000                BIC     #160000, R2     ;SAVE BANK BITS
5475  024774  060102                        ADD     R1,     R2      ;COMPUTE PHYSICAL ADDRESS
5476  024776  005503                        ADC     R3              ;MAKE SURE CARRY ISN'T LOST!
5477  025000  006302              1$:       ASL     R2              ;FIRST DIGIT TO R3
5478  025002  006103                        ROL     R3
5479  025004  012700  000006                MOV     #6,R0           ;DIGIT COUNT
5480  025010  000404                        BR      3$              ;PRINT FIRST DIGIT
5481  025012  006302              2$:       ASL     R2
5482  025014  006103                        ROL     R3
5483  025016  005301                        DEC     R1
5484  025020  001374                        BNE     2$
5485  025022  012701  000003      3$:       MOV     #3,R1           ;DIGIT SHIFT COUNT
5486  025026  062703  000060                ADD     #60,    R3      ;MAKE IT AN ASCII DIGIT
5487  025032  110367  000036                MOVB    R3,     8$      ;LOAD DIGIT INTO MESSAGE
5488  025036  004567  176176                JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
5489  025042  025074                        .WORD   8$              ;ADDRESS OF MESSAGE TO BE TYPED
5490  025044  005003                        CLR     R3              ;CLEAR INDEX
5491  025046  005300                        DEC     R0              ;DEC DIGIT COUNT
5492  025050  001360                        BNE     2$
5493  025052  012603                        MOV     (SP)+,R3        ;;POP STACK INTO R3
5494  025054  012602                        MOV     (SP)+,R2        ;;POP STACK INTO R2
5495  025056  012601                        MOV     (SP)+,R1        ;;POP STACK INTO R1
5496  025060  012600                        MOV     (SP)+,R0        ;;POP STACK INTO R0
5497  025062  012616                        MOV     (SP)+,  (SP)    ;ADJUST THE STACK TO CLEAR DATA
5498  025064  004567  176150                JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
5499  025070  026364                        .WORD   FILL2           ;ADDRESS OF MESSAGE TO BE TYPED
5500  025072  000207                        RTS     PC              ;RETURN
5501  025074     000              8$:       .BYTE   0               ;ONE DIGIT MESSAGE BUFFER
5502  025075     000                        .BYTE   0               ;MESSAGE TERMINATOR
5503
5504                                        .SBTTL  STANDARD PROGRAM MESSAGES
5505                                ;*****************************************************************
5506                                ;VARIOUS MESSAGE PRINTOUTS USED THRUOUT
5507                                ;THE PROGRAM
5508                                ;*****************************************************************
5509  025076  005015  052113  030461  MMAMES: .ASCIZ  <15><12>'KT11 (MEMORY MANAGEMENT) AVAILABLE'
5510  025104  024040  042515  047515
5511  025112  054522  046440  047101
5512  025120  043501  046505  047105
5513  025126  024524  040440  040526
5514  025134  046111  041101  042514
5515  025142     000
5516  025143     015  046412  046505  MEMMES: .ASCIZ  <15><12>'MEMORY MAP:'
5517  025150  051117  020131  040515
5518  025156  035120     000
5519  025161     015  041012  052131  BYTMES: .ASCIZ  <15><12>'BYTE MEMORY MAP:''
5520  025166  020105  042515  047515
5521  025174  054522  046440  050101
5522  025202  000072
5523  025204  005015  040520  044522  MTMAP:  .ASCIZ  <15><12>'PARITY MEMORY MAP:'
5524  025212  054524  046440  046505
5525  025220  051117  020131  040515
5526  025226  035120     000
5527  025231     015  043012  047522  FROM:   .ASCIZ  <15><12>'FROM '
```

```
5528  025236  020115     000
5529  025241     040  047524  000040  TO:      .ASCIZ  ' TO '
5530  025246  005015  047111  052523  INSUFF:  .ASCIZ  <15><12>'INSUFFICIENT MEMORY...FIRST 16K NOT ALL THERE!'
5531  025254  043106  041511  042511
5532  025262  052116  046440  046505
5533  025270  051117  027131  027056
5534  025276  044506  051522  020124
5535  025304  033061  020113  047516
5536  025312  020124  046101  020114
5537  025320  044124  051105  020505
5538  025326     000
5539  025327     015  047012  020117  MTR:     .ASCIZ  <15><12>'NO PARITY REGISTERS FOUND'
5540  025334  040520  044522  054524
5541  025342  051040  043505  051511
5542  025350  042524  051522  043040
5543  025356  052517  042116     000
5544  025363     015  051012  051505  PWRMSG:  .ASCIZ  <15><12>'RESTARTING AFTER A POWER FAILURE'<15><12>
5545  025370  040524  052122  047111
5546  025376  020107  043101  042524
5547  025404  020122  020101  047520
5548  025412  042527  020122  040506
5549  025420  046111  051125  006505
5550  025426  000012
5551  025430  005015  047516  050040  NOPES:   .ASCIZ  <15><12>'NO PARITY ERRORS FOUND ON MEMORY SCAN'<15><12>
5552  025436  051101  052111  020131
5553  025444  051105  047522  051522
5554  025452  043040  052517  042116
5555  025460  047440  020116  042515
5556  025466  047515  054522  051440
5557  025474  040503  006516  000012
5558  025502  005015  051120  043517  PROREL:  .ASCII  <15><12>'PROGRAM NOW RESIDES BACK AT 0 TO 8K'
5559  025510  040522  020115  047516
5560  025516  020127  042522  044523
5561  025524  042504  020123  040502
5562  025532  045503  040440  020124
5563  025540  020060  047524  034040
5564  025546     113
5565  025547     015  044012  052111           .ASCIZ  <15><12>'HIT CONTINUE FOR NORMAL RUNNING'<15><12>
5566  025554  041440  047117  044524
5567  025562  052516  020105  047506
5568  025570  020122  047516  046522
5569  025576  046101  051040  047125
5570  025604  044516  043516  005015
5571  025612     000
5572  025613     015  051012  043505  MX1:     .ASCIZ  <15><12>'REGISTER AT '
5573  025620  051511  042524  020122
5574  025626  052101  000040
5575  025632  041440  047117  051124  MX2:     .ASCIZ  ' CONTROLS '
5576  025640  046117  020123     000
5577  025645     015  047012  020117  NOMEM:   .ASCIZ  <15><12>'NO MEMORY FOUND.'
5578  025652  042515  047515  054522
5579  025660  043040  052517  042116
5580  025666  000056
5581  025670  005015  005012  047111  FADMES:  .ASCII  <15><12><12><12>'INPUT ALL PARAMETERS IN OCTAL.'
5582  025676  052520  020124  046101
5583  025704  020114  040520  040522
```

```
 5584  025712  042515  042524  051522
 5585  025720  044440  020116  041517
 5586  025726  040524  027114
 5587  025732  005015  044506  051522          .ASCIZ  <15><12>'FIRST ADDRESS:  '
 5588  025740  020124  042101  051104
 5589  025746  051505  035123  020040
 5590  025754     000
 5591  025755     015  046012  051501  LADMES: .ASCIZ  <15><12>'LAST ADDRESS:   '
 5592  025762  020124  042101  051104
 5593  025770  051505  035123  020040
 5594  025776  000040
 5595  026000  005015  040477  042104  BADADR: .ASCIZ  <15><12>'?ADDRESS IN UNMAPPED BANK?'
 5596  026006  042522  051523  044440
 5597  026014  020116  047125  040515
 5598  026022  050120  042105  041040
 5599  026030  047101  037513     000
 5600  026035     015  051412  046105  CONST:  .ASCIZ  <15><12>'SELECT CONSTANT:'
 5601  026042  041505  020124  047503
 5602  026050  051516  040524  052116
 5603  026056  000072
 5604  026060  005015  047125  054105  UNEXPT: .ASCIZ  <15><12>'UNEXPECTED MEMORY PARITY ERROR'
 5605  026066  042520  052103  042105
 5606  026074  046440  046505  051117
 5607  026102  020131  040520  044522
 5608  026110  054524  042440  051122
 5609  026116  051117     000
 5610  026121     015  050012  047522  PRELOC: .ASCIZ  <15><12>'PROGRAM RELOCATED TO '
 5611  026126  051107  046501  051040
 5612  026134  046105  041517  052101
 5613  026142  042105  052040  020117
 5614  026150     000
 5615  026151     015  046412  051117  MTOE:   .ASCIZ  <15><12>'MORE THAN ONE PARITY ERROR FOUND.'
 5616  026156  020105  044124  047101
 5617  026164  047440  042516  050040
 5618  026172  051101  052111  020131
 5619  026200  051105  047522  020122
 5620  026206  047506  047125  027104
 5621  026214     000
 5622  026215     015  051412  040503  SCANM:  .ASCIZ  <15><12>'SCANNING MEMORY FOR BAD PARITY.'
 5623  026222  047116  047111  020107
 5624  026230  042515  047515  054522
 5625  026236  043040  051117  041040
 5626  026244  042101  050040  051101
 5627  026252  052111  027131     000
 5628  026257     015  050012  051101  PEWNC:  .ASCIZ  <15><12>'PARITY ERROR WILL NOT CLEAR.'
 5629  026264  052111  020131  051105
 5630  026272  047522  020122  044527
 5631  026300  046114  047040  052117
 5632  026306  041440  042514  051101
 5633  026314  000056
 5634  026316  005015  047516  046440  NOMTST: .ASCIZ  <15><12>'NO MEMORY TESTED.'
 5635  026324  046505  051117  020131
 5636  026332  042524  052123  042105
 5637  026340  000056
 5638  026342  005015  045523  050111  SKPMES: .ASCIZ  <15><12>'SKIPPING TEST #'
 5639  026350  044520  043516  052040
```

```
5640  026356  051505  020124  000043
5641  026364  177777     000          FILL2:  .ASCIZ  <377><377>
5642
5643                                  .SBTTL  ERROR REPORTING MESSAGES AND TABLES.
5644                                  ;*****************************************************************
5645                                  ;* MESSAGE BLOCK FOR ERROR TABLE TYPEOUTS
5646                                  ;*****************************************************************
5647  026367     120  051101  052111  DM1:    .ASCIZ  'PARITY REGISTER DATA ERROR.'
5648  026374  020131  042522  044507
5649  026402  052123  051105  042040
5650  026410  052101  020101  051105
5651  026416  047522  027122     000
5652  026423     101  042104  042522  DM2:    .ASCIZ  'ADDRESS TEST ERROR(TST1-5).'
5653  026430  051523  052040  051505
5654  026436  020124  051105  047522
5655  026444  024122  051524  030524
5656  026452  032455  027051     000
5657  026457     103  047117  052123  DM4:    .ASCIZ  'CONSTANT DATA ERROR(TST6-10).'
5658  026464  047101  020124  040504
5659  026472  040524  042440  051122
5660  026500  051117  052050  052123
5661  026506  026466  030061  027051
5662  026514     000
5663  026515     122  052117  052101  DM5:    .ASCIZ  'ROTATING BIT ERROR(TST11-12).'
5664  026522  047111  020107  044502
5665  026530  020124  051105  047522
5666  026536  024122  051524  030524
5667  026544  026461  031061  027051
5668  026552     000
5669  026553     061  054040  051117  DM6:    .ASCIZ  '1 XOR 8 PATTERN ERROR(TST13).'
5670  026560  034040  050040  052101
5671  026566  042524  047122  042440
5672  026574  051122  051117  052050
5673  026602  052123  031461  027051
5674  026610     000
5675  026611     063  054040  051117  DM7:    .ASCIZ  '3 XOR 9 PATTERN ERROR(TST14-17).'
5676  026616  034440  050040  052101
5677  026624  042524  047122  042440
5678  026632  051122  051117  052050
5679  026640  052123  032061  030455
5680  026646  024467  000056
5681  026652  020070  047530  020122  DM10:   .ASCIZ  '8 XOR 13 PATTERN ERROR(TST20).'
5682  026660  031461  050040  052101
5683  026666  042524  047122  042440
5684  026674  051122  051117  052050
5685  026702  052123  030062  027051
5686  026710     000
5687  026711     120  051101  052111  DM11:   .ASCIZ  'PARITY MEMORY ADDRESS ERROR(TST21).'
5688  026716  020131  042515  047515
5689  026724  054522  040440  042104
5690  026732  042522  051523  042440
5691  026740  051122  051117  052050
5692  026746  052123  030462  027051
5693  026754     000
5694  026755     104  052101  050111  DM12:   .ASCIZ  "DATIP WITH WRONG PARITY DIDN'T TRAP(TST21)."
5695  026762  053440  052111  020110
```

```
5696  026770  051127  047117  020107
5697  026776  040520  044522  054524
5698  027004  042040  042111  023516
5699  027012  020124  051124  050101
5700  027020  052050  052123  030462
5701  027026  027051     000
5702  027031     127  047522  043516  DM13:   .ASCIZ  'WRONG PARITY TRAPPED, BUT NO REGISTER SHOWS ERROR FLAG.'
5703  027036  050040  051101  052111
5704  027044  020131  051124  050101
5705  027052  042520  026104  041040
5706  027060  052125  047040  020117
5707  027066  042522  044507  052123
5708  027074  051105  051440  047510
5709  027102  051527  042440  051122
5710  027110  051117  043040  040514
5711  027116  027107     000
5712  027121     120  051101  052111  DM14:   .ASCIZ  'PARITY REGISTER NOT MAPPED AS CONTROLLING THIS ADDRESS(TST21).'
5713  027126  020131  042522  044507
5714  027134  052123  051105  047040
5715  027142  052117  046440  050101
5716  027150  042520  020104  051501
5717  027156  041440  047117  051124
5718  027164  046117  044514  043516
5719  027172  052040  044510  020123
5720  027200  042101  051104  051505
5721  027206  024123  051524  031124
5722  027214  024461  000056
5723  027220  047515  042522  052040  DM16:   .ASCIZ  'MORE THAN ONE REGISTER INDICATED PARITY ERROR.'
5724  027226  040510  020116  047117
5725  027234  020105  042522  044507
5726  027242  052123  051105  044440
5727  027250  042116  041511  052101
5728  027256  042105  050040  051101
5729  027264  052111  020131  051105
5730  027272  047522  027122     000
5731  027277     104  052101  020101  DM17:   .ASCIZ  "DATA SHOULDN'T HAVE CHANGED WHEN PARITY ERROR TRAPPED(TST21)."
5732  027304  044123  052517  042114
5733  027312  023516  020124  040510
5734  027320  042526  041440  040510
5735  027326  043516  042105  053440
5736  027334  042510  020116  040520
5737  027342  044522  054524  042440
5738  027350  051122  051117  052040
5739  027356  040522  050120  042105
5740  027364  052050  052123  030462
5741  027372  027051     000
5742  027375     122  047101  047504  DM20:   .ASCIZ  'RANDOM DATA ERROR(TST22).'
5743  027402  020115  040504  040524
5744  027410  042440  051122  051117
5745  027416  052050  052123  031062
5746  027424  027051     000
5747  027427     111  051516  051124  DM21:   .ASCIZ  'INSTRUCTION EXECUTION ERROR(TST23-30).'
5748  027434  041525  044524  047117
5749  027442  042440  042530  052503
5750  027450  044524  047117  042440
5751  027456  051122  051117  052050
```

```
5752  027464  052123  031462  031455
5753  027472  024460  000056
5754  027476  041042  040522  041516  DM22:   .ASCIZ  '"BRANCH GOBBLE" ERROR(TST31).'
5755  027504  020110  047507  041102
5756  027512  042514  020042  051105
5757  027520  047522  024122  051524
5758  027526  031524  024461  000056
5759  027534  051120  043517  040522  DM23:   .ASCIZ  'PROGRAM CODE CHANGED WHEN RELOCATED.'
5760  027542  020115  047503  042504
5761  027550  041440  040510  043516
5762  027556  042105  053440  042510
5763  027564  020116  042522  047514
5764  027572  040503  042524  027104
5765  027600     000
5766  027601     124  040522  050120  DM24:   .ASCIZ  'TRAPPED, BUT NO REGISTER HAD ERROR BIT SET.'
5767  027606  042105  020054  052502
5768  027614  020124  047516  051040
5769  027622  043505  051511  042524
5770  027630  020122  040510  020104
5771  027636  051105  047522  020122
5772  027644  044502  020124  042523
5773  027652  027124     000
5774  027655     124  040522  050120  DM25:   .ASCIZ  'TRAPPED TO 114.'
5775  027662  042105  052040  020117
5776  027670  030461  027064     000
5777  027675     106  044501  042514  DM26:   .ASCIZ  'FAILED TO TRAP.'
5778  027702  020104  047524  052040
5779  027710  040522  027120     000
5780  027715     050  041501  044524  DM27:   .ASCIZ  "(ACTION ENABLE WASN'T SET)."
5781  027722  047117  042440  040516
5782  027730  046102  020105  040527
5783  027736  047123  052047  051440
5784  027744  052105  027051     000
5785  027751     015  052012  040522  DM31:   .ASCIZ  <15><12>'TRAPPED TO 4 '
5786  027756  050120  042105  052040
5787  027764  020117  020064     000
5788
5789                                  ;*****************************************************************
5790                                  ;DATA COLUMN HEADINGS
5791                                  ;*****************************************************************
5792
5793  027771     120  004503  042522  DH1:    .ASCIZ  'PC     REG     S/B     WAS'
5794  027776  004507  027523  004502
5795  030004  040527  000123
5796  030010  027526  041520  050011  DH2:    .ASCIZ  'V/PC   P/PC    MA      S/B     WAS'
5797  030016  050057  004503  040515
5798  030024  051411  041057  053411
5799  030032  051501     000
5800  030035     126  050057  004503  DH12:   .ASCIZ  'V/PC   P/PC    MA      S/B'
5801  030042  027520  041520  046411
5802  030050  004501  027523  000102
5803  030056  027526  041520  050011  DH14:   .ASCIZ  'V/PC   P/PC    REG     MA'
5804  030064  050057  004503  042522
5805  030072  004507  040515     000
5806  030077     126  050057  004503  DH15:   .ASCIZ  'V/PC   P/PC    MAUT    REG     S/B     WAS'
5807  030104  027520  041520  046411
```

```
5808  030112  052501  004524  042522
5809  030120  004507  027523  004502
5810  030126  040527  000123
5811  030132  027526  041520  050011  DH21:   .ASCIZ  'V/PC   P/PC    IUT     MA      S/B     WAS'
5812  030140  050057  004503  052511
5813  030146  004524  040515  051411
5814  030154  041057  053411  051501
5815  030162     000
5816  030163     126  050057  004503  DH22:   .ASCIZ  'V/PC   P/PC    PS S/B  PS WAS  MA      S/B     WAS'
5817  030170  027520  041520  050011
5818  030176  020123  027523  004502
5819  030204  051520  053440  051501
5820  030212  046411  004501  027523
5821  030220  004502  040527  000123
5822  030226  027526  041520  050011  DH23:   .ASCIZ  'V/PC   P/PC    SRC MA  DST MA  S/B     WAS'
5823  030234  050057  004503  051123
5824  030242  020103  040515  042011
5825  030250  052123  046440  004501
5826  030256  027523  004502  040527
5827  030264  000123
5828  030266  027526  041520  050011  DH24:   .ASCIZ  'V/PC   P/PC    TRP/PC'
5829  030274  050057  004503  051124
5830  030302  027520  041520     000
5831  030307     126  050057  004503  DH25:   .ASCIZ  'V/PC   P/PC    TRP/PC  REG     WAS'
5832  030314  027520  041520  052011
5833  030322  050122  050057  004503
5834  030330  042522  004507  040527
5835  030336  000123
5836  030340  027526  041520  050011  DH26:   .ASCIZ  'V/PC   P/PC    REG     WAS'
5837  030346  050057  004503  042522
5838  030354  004507  040527  000123
5839  030362  042522  004507  040527  DH30:   .ASCIZ  'REG    WAS     MA      WAS'
5840  030370  004523  040515  053411
5841  030376  051501     000
5842
5843                                   ;*****************************************************************
5844                                   ;* DATA FORMAT TABLE FOR ERROR PRINTOUT.
5845                                   ;*****************************************************************
5846  030401     000     377     000  DF1:    .BYTE   0,-1,0,0
5847  030404     000
5848  030405     000     377     377  DF2:    .BYTE   0,-1,-1,0,0
5849  030410     000     000
5850  030412     000     377     377  DF3:    .BYTE   0,-1,-1,-2,-2
5851  030415     376     376
5852  030417     000     377     377  DF14:   .BYTE   0,-1,-1,-1,0,0
5853  030422     377     000     000
5854  030425     000     377     000  DF21:   .BYTE   0,-1,0,-1,0,0
5855  030430     377     000     000
5856  030433     000     377     376  DF22:   .BYTE   0,-1,-2,-2,-1,0,0
5857  030436     376     377     000
5858  030441     000
5859  030442     377     000     377  DF30:   .BYTE   -1,0,-1,-2
5860  030445     376
5861                                           .EVEN
5862
5863          032110                   . = 32110                    ;THE LOADERS ARE SAVE HERE TO END OF 8K
```

```
5864
5865            000001                          .END
```

```
ABASE = 000000            968    1015
ACDW1 = 000000            968    1017
ACDW2 = 000000            968    1018
ACPUOP= 000000            968     986
ADDW0 = 000000            968    1019
ADDW1 = 000000            968    1020
ADDW10= 000000            968    1029
ADDW11= 000000            968    1030
ADDW12= 000000            968    1031
ADDW13= 000000            968    1032
ADDW14= 000000            968    1033
ADDW15= 000000            968    1034
ADDW2 = 000000            968    1021
ADDW3 = 000000            968    1022
ADDW4 = 000000            968    1023
ADDW5 = 000000            968    1024
ADDW6 = 000000            968    1025
ADDW7 = 000000            968    1026
ADDW8 = 000000            968    1027
ADDW9 = 000000            968    1028
ADEVCT= 000000            968     977
ADEVM = 000000            968    1016
AE    = 000001            768#   3177    3262    4409
AENV  = 000000            968     982
AENVM = 000000            968     983
AFATAL= 000000            968     974
AMADR1= 000000            968     999
AMADR2= 000000            968    1003
AMADR3= 000000            968    1006
AMADR4= 000000            968    1009
AMAMS1= 000000            968     993
AMAMS2= 000000            968    1001
AMAMS3= 000000            968    1004
AMAMS4= 000000            968    1007
AMSGAD= 000000            968     979
AMSGLG= 000000            968     980
AMSGTY= 000000            968     973
AMTYP1= 000000            968     994
AMTYP2= 000000            968    1002
AMTYP3= 000000            968    1005
AMTYP4= 000000            968    1008
APASS = 000000            968     976
APRIOR= 000000            968    1012
APTCSU= 000040           5140    5284#
APTENV= 000001           4836    5133    5203    5282#
APTSIZ= 000200           1447    5281#
APTSPO= 000100           5135    5205    5283#
ASWREG= 000000            968     984
ATESTN= 000000            968     975
AUNIT = 000000            968     978
AUSWR = 000000            968     985
AVECT1= 000000            968    1010
AVECT2= 000000            968    1011
BADADR  026000           1927    5595#
BANKNO  016432           2114    2123    2150    2160    4075#
BGEND   014174           3665    3691    3693#
```

```
BGERR   014176          1151    3696#
BGEXIT  014310          1152    3716#
BGLEN = 000025          3648    3651    3661    3694#
BGLOOP  014036          3649#   3721
BGTOP   014032          3648#
BG1     014126          3674#
BG2     014140          3678#   3682
BG3     014152          3674    3683#   3699
BG4     014153          3684#
BG5     014154          3680    3685#
BG6     014160          3685    3687#
BG7     014172          3689    3690    3692#   3705
BITPT   001542          1112#   1753*   1754*   1782    1783    1784    1785    1793*   1794*   1796    1798    1801    1804*
                        3154    3156    3211    3213    3826*   3827*   3834    3836    3839*   3840*   3843    3845    3850
                        3851    3870    3873*   3876    3877    3891    3892    3893    3895    3904*   3905*   3909*   3914
                        3916    3937*   3938*   3940    3942    3944    3946    3950    3951    3981*   3983    3985    3988
                        3995    3997    4002    4003    4018    4020    4028*   4029*   4031    4033    4038*   4040    4042
                        4044    4078    4079
BIT0  = 000001           710#    823     839    1594    1817    1850    1861    1951    1969    3200    3229    3244    3826
                        4426    4459    4590    4729
BIT00 = 000001           700#    710
BIT01 = 000002           699#    709
BIT02 = 000004           698#    708
BIT03 = 000010           697#    707
BIT04 = 000020           696#    706
BIT05 = 000040           695#    705
BIT06 = 000100           694#    704
BIT07 = 000200           693#    703
BIT08 = 000400           692#    702    4678
BIT09 = 001000           691#    701    4688    4846
BIT1  = 000002           709#
BIT10 = 002000           690#   4822
BIT11 = 004000           689#   4695
BIT12 = 010000           688#
BIT13 = 020000           687#   3852    3879    3952    3989    4830
BIT14 = 040000           686#    805    4664
BIT15 = 100000           685#   3229    3904    4205
BIT2  = 000004           708#
BIT3  = 000010           707#
BIT4  = 000020           706#
BIT5  = 000040           705#
BIT6  = 000100           704#
BIT7  = 000200           703#
BIT8  = 000400           702#   3909    4238
BIT9  = 001000           701#
BLKMSK  001604          1137#   3090    3091*   3101*   3829    3900    3932    4715*   4720*
BPTVEC= 000014           717#
BRGOB   014124          1150    3673#   3694
BRGOBL  014026          3646#
BYTMES  025161          1588    5519#
CKPMER  020054          2241    3162    4009    4418#
CLRPAR  020506          1752    1812    4463    4534#
CONST   026035          1939    5600#
CTRLS   004102          1668    1682#
DDISP = 177570           632#    945
DF1     030401          1296    5846#
```

```
DF14    030417          1351    1356    1361    1387    1392    1397    5852#
DF2     030405          1301    1311    1316    1321    1326    1331    1372    1402    1407    5848#
DF21    030425          1377    5854#
DF22    030433          1382    5856#
DF3     030412          1306    1336    1341    1346    1367    5850#
DF30    030442          1412    1417    5859#
DH1     027771          1294    5793#
DH12    030035          1339    1344    5800#
DH14    030056          1349    1359    5803#
DH15    030077          1354    5806#
DH2     030010          1299    1304    1309    1314    1319    1324    1329    1334    1365    1370    5796#
DH21    030132          1375    5811#
DH22    030163          1380    5816#
DH23    030226          1385    5822#
DH24    030266          1390    5828#
DH25    030307          1395    5831#
DH26    030340          1400    1405    5836#
DH30    030362          1410    5839#
DIDBH   013420          3458#
DIDBL   013320          3408#
DIDO    013222          3358#
DIPDO   013524          3509#
DISPLA  001140           945#   1442*   1452*   4272*   4325*   4710*   4821*
DISPRE  000174           782#   1442
DM1     026367          1293    1353    5647#
DM10    026652          1328    5681#
DM11    026711          1333    5687#
DM12    026755          1338    5694#
DM13    027031          1343    5702#
DM14    027121          1348    5712#
DM16    027220          1358    5723#
DM17    027277          1363    5731#
DM2     026423          1298    1303    5652#
DM20    027375          1369    5742#
DM21    027427          1374    5747#
DM22    027476          1379    5754#
DM23    027534          1384    5759#
DM24    027601          1389    5766#
DM25    027655          1394    5774#
DM26    027675          1399    5777#
DM27    027715          1404    5780#
DM31    027751          1414    5785#
DM4     026457          1308    5657#
DM5     026515          1313    5663#
DM6     026553          1318    5669#
DM7     026611          1323    5675#
DONE    014340          3727#
DPDBH   013724          3609#
DPDBL   013624          3559#
DSWR  = 177570           631#    944
DT1     001642          1162#   1295
DT12    001670          1166#   1340    1345
DT14    001702          1168#   1350    1360
DT15    001714          1170#   1355
DT2     001654          1164#   1300    1305    1310    1315    1320    1325    1330    1335    1366    1371
DT21    001732          1173#   1376
```

DZQMCB.P11 CROSS REFERENCE TABLE -- USER SYMBOLS

```
DT22    001750          1176*   1381
DT23    001770          1179*   1386
DT24    002006          1182*   1391
DT25    002016          1184*   1396
DT26    002C32          1186*   1401    1406
DT30    002044          1188*   1411
DT31    002056          1190*   1416
EMTVEC= 000030           720*
ENDINS  022032          4714    4787    4800*
ERRTRP  024636           794    1477    1559    1666    5435*
ERRVEC= 000004           713*    793    1473*   1477*   1489*   1519*   1559*   1582*   1592*   1652*   1666*   4266*   4319*
                        4669    4670*   4672*   4675*
FADMAP  001566          1130*   1902*   1903*   3893    3895    4018    4020    4042    4044    4729*   4756    4757    4765
                        4767
FADMES  025670          1855    5581*
FADMSK  001564          1128*   4046    4790*   4792*
FILL2   026364          5499    5641*
FLAG8K  001555          1122*   3885*   3975*   3978*   3996*
FROM    025231          1601    4610    5527*
FSTADR  001560          1125*   1838    1936*   4403    4721
GMPR    003766          1568    1648*
GMPRA   004014          1653*   1664
GMPRB   004030          1652    1659*
GMPRC   004036          1657    1661*
GMPRD   004074          1650    1672*
GNS   = ****** U         781    1458
INITDN  015366          2007    2044    2072    2149    2159    3900*
INITEX  015500          3896    3898    3917    3919*
INITMM  014730          1990    2034    2083    2113    2122    2192    2207    2244    2266    2290    2316    2350    2395
                        2411    2464    2474    2517    2567    2578    2621    2671    2682    2730    2857    2868    2916
                        3041    3059    3092    3107    3153    3299    3361    3411    3461    3512    3562    3612    3647
                        3826*   4095
INSERT  021332          4712*
INSUFF  025246          1570    5530*
IOTVEC= 000020           718*
KIPAR0= 172340           754*    830     849*    3811*   4232*   4298*   5468
KIPAR1= 172342           755*    855*    3812*   4233*   4299*
KIPAR2= 172344           756*   1530    1550*   1627*   1792*   3228    3813*   3832*   3838*   3849    3906*   3936*   3954
                        4027*   4059    4214*   4217*   4232    4293*   4468    4469*   4508*   4513*
KIPAR3= 172346           757*   1624    1627    1628*   3814*   3849*   3854*   3954*   3955*   4203*   4206*   4214    4233
                        4294*
KIPAR4= 172350           758*   3815*
KIPAR5= 172352           759*   3816*
KIPAR6= 172354           760*   3817*
KIPAR7= 172356           761*   3818*
KIPDR0= 172300           743*    825    3803*
KIPDR1= 172302           744*   3804*
KIPDR2= 172304           745*   3805*
KIPDR3= 172306           746*   3806*
KIPDR4= 172310           747*   3807*
KIPDR5= 172312           748*   3808*
KIPDR6= 172314           749*   3809*
KIPDR7= 172316           750*   3810*
LADMAP  001600          1136*   1930*   1931*   1954*   1955*   3843    3845    3862    3864    3877    3886    3914    3916
                        3928    3930    3944    3946    3963    3965    3985    3995    3997    4763    4764    4765    4767
LADMES  025755          1882    5591*
```

```
LADMSK  001576  1134# 3847  3866  3888  3948  3967  3987  3999  4795* 4797*
LDDISP  001520  1098# 1451*
LMAD    001516  1097# 1469  1483* 4337
LSTADR  001572  1131# 1844* 1894* 1898  1909  1948* 1949* 1950* 1951* 1952  3918  4730  4737
MAMF    017754  1966  3289  4395#
MAMF1   020040  4409# 4411
MAMF2   020052  4396  4398  4404  4413#
MANUAL  004646  1673  1837# 1929
MANUL1  004704  1843  1850#
MANUL2  005306  1845  1946#
MAPMEM  003220  1488  1489  1507#
MAPRB   004344  1757  1767#
MASK4K= 017777   772# 1521  1546  1620  1900  1952  3091  3890  4001  4501  4754  4761  4790
                4795
MEMMAP  001522   818  1099# 1508  1509  1796  1798  1801  1875  1877  1916  1918  1921  1923
                4210  4212  4221  4223  4243  4248  4471  4473
MEMMES  025143  1517  5516#
MMAMES  025076  1494  5509#
MMAVA   000606   821   878# 1486* 1492* 1527  1548  1579* 1585* 1622  1756  1790  1932  1946
                1967  3225  3830  3902  3933  4025  4056  4201  4291  4340  4348  4466  4506
                5457
MMDOWN  016176  2019  2056  2079  2155  2171  3912  4018#
MMINIT  014602  1491  1584  1759  1971  3802#
MMORE   001552  1118# 3910* 3911* 3919* 4004  4047
MMUP    015506  2003  2040  2097  2118  2133  2196  2218  2255  2279  2303  2344  2390  2405
                2451  2468  2511  2554  2571  2615  2658  2675  2723  2843  2861  2909  3029
                3053  3085  3100  3133  3288  3324  3375  3425  3476  3526  3576  3627  3725
                3928# 4099
MMVEC = 000250    732#
MPEND   002110  1209#
MPPATS  002064  1154  1198#
MPRX    002252  1149  1273# 1648  1663  1667  1682  1722  1727  1787  1834  3146  4007  4395
                4418
MPR0    002112  1148  1224# 1651  1769  1816
MPR1    002120  1227#
MPR10   002206  1254#
MPR11   002214  1257#
MPR12   002222  1260#
MPR13   002230  1263#
MPR14   002236  1266#
MPR15   002244  1269#
MPR2    002126  1230#
MPR3    002134  1233#
MPR4    002142  1236#
MPR5    002150  1239#
MPR6    002156  1242#
MPR7    002164  1245#
MPR8    002172  1248#
MPR9    002200  1251#
MTMAP   025204  1814  5523#
MTOE    026151  4378  5615#
MTR     025327  1670  5539#
MX1     025613  1820  5572#
MX2     025632  1828  5575#
NOMEM   025645  4582  5577#
NOMTST  026316  4733  5634#
```

```
NOPES   025430  4520   5551#
OEFLG   001556  1123#  3173*  3281*
PARVEC= 000114   769#  3186*  3190*  3197*  4268*  4321*  4405*  4406*
PBTRP   012456  1153   3197#
PC     =%000007   644#   816    864*   870*   912    1460   1491*  1560*  1584*  1604*  1615*  1637*  1689*
                 1690*  1702*  1703*  1709*  1710*  1734*  1735*  1752*  1759*  1612*  1826*  1832*  1860*
                 1887*  1944*  1966*  1971*  1991*  1996*  1997*  2003*  2008*  2013*  2014*  2019*  2035*
                 2040*  2045*  2050*  2051*  2056*  2073*  2079*  2084*  2090*  2091*  2097*  2114*  2118*
                 2123*  2127*  2128*  2133*  2150*  2155*  2160*  2165*  2166*  2171*  2196*  2212*  2213*
                 2218*  2241*  2249*  2250*  2255*  2265*  2268*  2273*  2274*  2279*  2289*  2292*  2297*
                 2298*  2303*  2344*  2356*  2357*  2364*  2365*  2372*  2373*  2380*  2381*  2390*  2405*
                 2417*  2418*  2425*  2426*  2433*  2434*  2441*  2442*  2451*  2465*  2468*  2480*  2481*
                 2487*  2488*  2494*  2495*  2501*  2502*  2511*  2524*  2525*  2533*  2534*  2542*  2543*
                 2554*  2568*  2571*  2584*  2585*  2591*  2592*  2598*  2599*  2605*  2606*  2615*  2628*
                 2629*  2637*  2638*  2646*  2647*  2658*  2672*  2675*  2688*  2689*  2695*  2696*  2702*
                 2703*  2709*  2710*  2723*  2736*  2737*  2745*  2746*  2754*  2755*  2761*  2762*  2770*
                 2771*  2779*  2780*  2786*  2787*  2795*  2796*  2804*  2805*  2811*  2812*  2820*  2821*
                 2829*  2830*  2843*  2858*  2861*  2874*  2875*  2881*  2882*  2888*  2889*  2895*  2896*
                 2909*  2922*  2923*  2931*  2932*  2940*  2941*  2947*  2948*  2956*  2957*  2965*  2966*
                 2972*  2973*  2981*  2982*  2990*  2991*  2997*  2998*  3006*  3007*  3015*  3016*  3029*
                 3053*  3068*  3069*  3075*  3076*  3085*  3100*  3116*  3117*  3123*  3124*  3133*  3152*
                 3161*  3162*  3169*  3170*  3191*  3192*  3207*  3208*  3215*  3216*  3235*  3236*  3248*
                 3249*  3265*  3266*  3274*  3275*  3288*  3289*  3297   3314*  3315*  3324*  3368*  3369*
                 3375*  3418*  3419*  3425*  3469*  3470*  3476*  3519*  3520*  3526*  3569*  3570*  3576*
                 3620*  3621*  3627*  3673   3711*  3725*  3742*  3745*  3746*  3763*  3766*  3775*  3783*
                 3820*  3912*  4005*  4009*  4010*  4048*  4070*  4089*  4099*  4100*  4122*  4154*  4176*
                 4187*  4262*  4283*  4308*  4330*  4351*  4369*  4381*  4382*  4384*  4413*  4428*  4429*
                 4432*  4434*  4435*  4437*  4441*  4463*  4486*  4487*  4529*  4543*  4571*  4622*  4630*
                 4644*  4736*  4784*  4800*  4832*  4839*  4853*  4875*  4908*  4915*  4920*  4928*  4941*
                 4976*  5059*  5105*  5138*  5156*  5163*  5170*  5184*  5186*  5222*  5254*  5266*  5276*
                 5435   5440*  5500*
PESRV   017646  1155   4358#
PEWNC   026257  4495   5628#
PHYADR  016356  1991   2008   2035   2045   2073   2084   4054#
PIRQ   = 177772   630#
PIRQVE= 000240   724#
PMEMAP  001536  1109#  1767*  1768*  1784*  1785*  3154   3156
PRELOC  026121  4184   5610#
PRGMAP  000602   841    842    862    877#   1464*  1465*  3732   3734   3738   4195   4227   4235*  4253
                 4279*  4288   4328*  4329*  4740   4741
PROREL  025502  5558#
PR0    = 000000   647#
PR1    = 000040   648#
PR2    = 000100   649#
PR3    = 000140   650#
PR4    = 000200   651#
PR5    = 000240   652#
PR6    = 000300   653#
PR7    = 000340   654#
PS     = 177776   627#   628
PSCAN   020162  4384   4432   4437   4448#
PSW    = 177776   628#  1825   1859   1886   1943   3696   3774   4621   4629   4783   4874   4907   4914
                 4927   4975   5058   5104
PWRMSG  025363   911   5544#
PWRVEC= 000024   719#   884*   885*   893*   908*   909*  1434*  1435*  4267*  4320*
RADTAB  001610  1145#  4273   4312
```

```
RANTST  013074          3297#
RELOC   016640          4159#  4229   4258   4295   4305
RELOCF  000600           857    876#  1466*  1965   3911   4261*  4265*  4274   4282   4303   4327*  4399   4722
                        4878   4899   4901   5101   5309
RELTOP  016762          3742   4195#
RELO    017364           864   3745   4288#
RESCHK  004230          1722#
RESLDR  017572           870   3746   4337#
RESRVD  001514          1095#  1147   1687   1692   1697   1707   3230   3231   3260
RESTAR  000300           789    814#   913   1463
RESTOR  000304           790    816#
REST1   000306           815    817#
REST2   000324           819    821#
RESVEC= 000010           714#
ROTATE  016512          2268   2292   4104#
RW    = 000006           764#  3803   3804   3805   3806   3810
RO    =%000000           635#   825*   827*   830*   831*   832*   833*   834*   835*   836*   837*   838*   840*
                         846*   849    851*   855    857*   858*   859    860*   861    886    907*  1460*  1461
                        1471*  1472   1474*  1478   1480   1508*  1510*  1523*  1563   1565*  1566*  1567   1590*
                        1598   1608   1617*  1630   1684*  1692   1694   1700   1712*  1830*  1831   1837*  1840*
                        1850*  1871*  1875   1902   1904   1913*  1916   1921   1930   1992   1994   2000*  2009*
                        2011   2036   2037*  2046*  2048   2074*  2075*  2076   2085*  2088   2094*  2115   2125
                        2151*  2152   2161*  2163   2191*  2193   2206*  2210   2242*  2245   2247   2264*  2271
                        2288*  2295   2314*  2317*  2320   2322   2325   2327   2330   2332   2335   2337   2349*
                        2354   2360*  2362   2368*  2370   2376*  2378   2384*  2387*  2410*  2415   2421*  2423
                        2429*  2431   2437*  2439   2445*  2448*  2462*  2473*  2478   2485   2492   2499   2505*
                        2508*  2516*  2522   2528*  2531   2537*  2540   2548*  2551*  2565*  2577*  2582   2589
                        2596   2603   2609*  2612*  2620*  2626   2632*  2635   2641*  2644   2652*  2655*  2669*
                        2680*  2686   2693   2700   2707   2713   2714*  2718   2719*  2728*  2734   2740*  2743
                        2749*  2752   2759   2765*  2768   2774*  2777   2784   2790*  2793   2799*  2802   2809
                        2815*  2818   2824*  2827   2833   2834*  2838   2839*  2855*  2866*  2872   2879   2886
                        2893   2899   2900*  2904   2905*  2914*  2920   2926*  2929   2935*  2938   2945   2951*
                        2954   2960*  2963   2970   2976*  2979   2985*  2988   2995   3001*  3004   3010*  3013
                        3019   3020*  3024   3025*  3040*  3043*  3045*  3046   3058*  3061*  3063*  3066   3073
                        3106*  3109*  3111*  3114   3121   3151*  3174*  3175   3181   3219   3220*  3221*  3222*
                        3223*  3224*  3227*  3228*  3229*  3230*  3233   3262*  3263   3271*  3272   3278   3280*
                        3310*  3312   3360*  3366   3410*  3416   3460*  3467   3511*  3517   3561*  3567   3611*
                        3618   3650*  3652   3667*  3686   3692   3780*  3783   4054*  4068*  4075*  4083*  4096
                        4129   4130   4131   4132   4139   4140   4141   4142   4151   4152*  4199   4204*  4208*
                        4212   4215   4219*  4223   4226*  4227   4238*  4241*  4243   4246*  4248   4250   4251*
                        4252*  4253   4279   4281*  4311   4312*  4313*  4314   4316   4318*  4337*  4345*  4449
                        4459*  4471   4510*  4515*  4528*  4569   4577   4579   4594   4596   4865   4866*  4867*
                        4879*  4880*  4881*  4882*  4883*  4884   4890   4897   4900*  4901*  4902   4917   4938*
                        5052   5060*  5061   5064   5084*  5087*  5131   5132*  5137   5142   5145*  5199   5207*
                        5211   5212   5214*  5215*  5216   5227*  5232   5238   5253*  5257   5258*  5259*  5262
                        5265*  5268   5275*  5299   5309*  5311   5314   5330   5331   5344*  5451   5470*  5473*
                        5479*  5491*  5496*
R1      =%000001         636#   826*   828*   841*   844*   853*   887    906*  1472*  1480*  1509*  1511*  1524*
                        1564   1686*  1687*  1695*  1697*  1700   1706*  1707*  1729*  1731*  1732   1838*  1839*
                        1851*  1872*  1877   1903   1905   1914*  1918   1923   1931   1993*  1994   2010*  2011
                        2047*  2048   2087*  2088   2124*  2125   2162*  2163   2209*  2210   2246*  2247   2269*
                        2271   2293*  2295   2353*  2354   2361*  2362   2369*  2370   2377*  2378   2414*  2415
                        2422*  2423   2430*  2431   2438*  2439   2477*  2478   2484*  2485   2491*  2492   2498*
                        2499   2521*  2522   2530*  2531   2539*  2540   2581*  2582   2588*  2589   2595*  2596
                        2602*  2603   2625*  2626   2634*  2635   2643*  2644   2685*  2686   2692*  2693   2699*
                        2700   2706*  2707   2733*  2734   2742*  2743   2751*  2752   2758*  2759   2767*  2768
```

```
                    2776* 2777  2783* 2784  2792* 2793  2801* 2802  2808* 2809  2817* 2818  2826*
                    2827  2871* 2872  2878* 2879  2885* 2886  2892* 2893  2919* 2920  2928* 2929
                    2937* 2938  2944* 2945  2953* 2954  2962* 2963  2969* 2970  2978* 2979  2987*
                    2988  2994* 2995  3003* 3004  3012* 3013  3065* 3066  3072* 3073  3113* 3114
                    3120* 3121  3167* 3202* 3231* 3233  3246* 3259* 3260* 3263  3270* 3272  3311*
                    3312  3365* 3366  3415* 3416  3466* 3467  3516* 3517  3566* 3567  3617* 3618
                    3651* 3655* 3659* 3661* 3663* 3668* 3693  4058  4059* 4060* 4061* 4062* 4063*
                    4064* 4066* 4068  4069* 4076  4078* 4081* 4088* 4200  4205* 4207* 4210  4216
                    4218* 4221  4225* 4235  4239* 4240* 4245* 4257  4261  4262  4264  4265  4266
                    4267  4268  4271  4272  4273* 4274* 4275  4277  4280* 4343* 4345  4362  4388*
                    4450  4460* 4473  4476  4477* 4503* 4511* 4527* 4570  4586  4590* 4594  4637*
                    4643* 4896  4897* 4899* 4904  4911  4919  4924  4931  4937* 5053  5062* 5070*
                    5072* 5074* 5077* 5080  5083* 5200  5229* 5230  5232  5234  5236  5238* 5239*
                    5252* 5269* 5270  5272* 5300  5314* 5315  5319  5343* 5452  5461* 5462* 5463*
                    5464* 5465* 5466* 5467* 5468* 5469* 5471* 5475  5483* 5485* 5495*
R2      =%000002     637#  842*  843*  852*  888   905* 1479* 1481* 1512* 1520* 1521  1525  1546*
                    1547* 1551* 1591* 1593  1594  1596* 1597* 1603  1613* 1614  1616* 1618* 1620
                    1626* 1635* 1636  1683* 1685* 1686  1694* 1695  1696* 1706  1728* 1729  1730*
                    1755* 1758* 1774* 1775  1789* 1803* 1852* 1864* 1869  1909  1911* 1992* 1993
                    2001  2010  2017  2036* 2038  2047  2054  2076* 2077  2087  2095  2115* 2116
                    2124  2131  2152* 2153  2162  2169  2193* 2194  2209  2216  2245* 2246  2253
                    2269  2277  2293  2301  2320* 2321* 2322* 2323* 2325* 2326* 2327* 2328* 2330*
                    2331* 2332* 2333* 2335* 2336* 2337* 2338* 2342  2353  2361  2369  2377  2388
                    2397* 2398* 2399* 2400* 2403  2414  2422  2430  2438  2449  2466  2477  2484
                    2491  2498  2509  2521  2529* 2530  2538* 2539  2552  2569  2581  2589  2595
                    2602  2613  2625  2633* 2634  2642* 2643  2656  2673  2685  2692  2699  2706
                    2721  2733  2741* 2742  2750* 2751  2758  2766* 2767  2775* 2776  2783  2791*
                    2792  2800* 2801  2808  2816* 2817  2825* 2826  2841  2859  2871  2878  2885
                    2892  2907  2919  2927* 2928  2936* 2937  2944  2952* 2953  2961* 2962  2969
                    2977* 2978  2986* 2987  2994  3002* 3003  3011* 3012  3027  3046* 3051  3065
                    3072  3083  3094* 3095* 3098  3113  3120  3131  3158* 3159* 3163  3165* 3167
                    3175* 3181* 3188* 3220  3252  3255* 3278* 3279  3284* 3285  3300  3302* 3306
                    3309* 3311  3322  3362* 3363* 3364  3365  3372* 3373  3412* 3413* 3414  3415
                    3422* 3423  3462* 3463* 3464  3465* 3466  3473* 3474  3513* 3514* 3515  3516
                    3523* 3524  3563* 3564* 3565  3566  3573* 3574  3613* 3614* 3615  3616* 3617
                    3624* 3625  3649  3652* 3653  3657  3662* 3665* 3666* 3669  3716* 3717  3723*
                    3724* 3828* 3833* 3872* 3897* 3901* 3918* 3935* 3971  3979* 3980* 4022  4035*
                    4037* 4054  4077  4079* 4080* 4087* 4096* 4097  4104* 4105* 4106* 4107* 4108*
                    4109* 4110* 4111* 4112* 4113* 4114* 4115* 4116* 4117* 4118* 4119* 4120* 4121*
                    4129* 4130* 4131* 4132* 4134* 4135* 4136* 4137* 4139* 4140* 4141* 4142* 4144*
                    4145* 4146* 4147* 4160  4163* 4166  4170  4172  4174  4190* 4344* 4346* 4451
                    4461* 4470* 4477  4489* 4491  4500* 4501  4505* 4509* 4526* 4550  4559  4562
                    4567  4587  4591* 4596  4638* 4642* 5054  5063* 5071* 5073* 5075* 5081  5082*
                    5301  5313* 5317* 5320  5327* 5328* 5329  5334* 5342* 5453  5455* 5461  5474*
                    5475* 5477* 5481* 5494*
R3      =%000003     638#  889   904* 1513* 1523  1552* 1557* 1651* 1653* 1654  1656  1660* 1661*
                    1662* 1663  1682* 1683  1714  1722* 1723* 1724  1727* 1728  1738  1769* 1770
                    1772* 1776* 1777  1782* 1783* 1786* 1787  1816* 1817  1822  1830  1833* 1834
                    1853* 1865* 1866  1906  1912* 2315* 2318* 2321  2323  2326  2328  2331  2333
                    2336  2338  2463* 2519* 2546* 2566* 2623* 2650* 2670* 2681* 2714  2715* 2719
                    2720* 2729* 2834  2835* 2839  2840* 2856* 2867* 2900  2901* 2905  2906* 2915*
                    3020  3021* 3025  3026* 3044* 3047* 3062* 3079* 3110* 3127* 3176* 3177* 3178
                    3182* 3183* 3184  3199* 3200  3202  3204* 3205  3211  3213  3239* 3240  3241*
                    3242  3244  3246  3258* 3259  3269* 3297* 3298* 3301  3302  3303  3305* 3308*
                    3310  3318  3320* 3358* 3362  3372  3408* 3412  3422  3458* 3462  3473  3509*
                    3513  3523  3559* 3563  3573  3609* 3613  3624  3673* 3674* 3675  3677* 3679*
```

```
                    3698   3703   3704   4134   4135   4136   4137   4144   4145   4146   4147   4152   4153*
                    4161   4164*  4166*  4170   4173   4175   4186   4189*  4363   4364*  4365   4367   4373
                    4375   4387*  4407   4408*  4409*  4410   4412*  4422   4423*  4424   4426   4438   4440*
                    4452   4478*  4479   4490*  4492   4497*  4498   4525*  4535   4536*  4537   4539*  4542*
                    4554   4555   4558   4566   4588   4592*  4593   4598   4600*  4605   4607*  4612   4614*
                    4616*  4625   4634*  4635*  4641*  4968   4970*  4971   4977*  4978   4986*  4987   4989
                    4999   5004   5006*  5016   5019   5021*  5025*  5302   5310*  5311*  5312*  5326*  5329*
                    5338*  5339*  5341*  5389   5398*  5404*  5405*  5408*  5413*  5414*  5415   5425*  5454
                    5456*  5472*  5476*  5478*  5482*  5486*  5487   5490*  5493*
R4      =%000004     639#    890    903*  1514*  1524   1553*  1648*  1656*  1665*  1672*  1879*  1895   1990*
                    2007*  2034*  2044*  2072*  2083*  2113*  2122*  2149*  2159*  2192*  2207*  2240*  2242
                    2244*  2266*  2290*  2316*  2319*  2340*  2350*  2351*  2385*  2395*  2396*  2401*  2411*
                    2412*  2446*  2464*  2474*  2475*  2506*  2517*  2518*  2549*  2567*  2578*  2579*  2610*
                    2621*  2622*  2653*  2671*  2682*  2683*  2716*  2730*  2731*  2836*  2857*  2868*  2869*
                    2902*  2916*  2917*  3022*  3041*  3042*  3049*  3059*  3060*  3081*  3092*  3093*  3096*
                    3107*  3108*  3129*  3153*  3252*  3270   3299*  3359*  3360   3361*  3363   3409*  3411*
                    3413   3459*  3461*  3463   3510*  3512*  3514   3560*  3562*  3564   3610*  3612*  3614
                    3647*  3675*  3676*  3681*  3688*  3919   3920*  4095*  4127*  4149*  4162   4165*  4167*
                    4169*  4181*  4188*  4453   4462*  4481   4484*  4517   4524*  4589   4593*  4601*  4608*
                    4613   4615*  4617   4633*  4636*  4640*  4662*  4748   4750   4752   4758   4760   5390
                    5392*  5393*  5394*  5395   5396*  5410   5412*  5421*  5424*
R5      =%000005     640#    814*   816*   865    891    902*   910*  1455*  1493*  1516*  1530*  1532*  1533*
                    1534*  1535*  1536*  1538*  1540   1561*  1569*  1587*  1600*  1610*  1632*  1669*  1813*
                    1819*  1827*  1854*  1881*  1926*  1938*  1983*  2001   2017   2030*  2038   2054   2068*
                    2077   2095   2109*  2116   2131   2145*  2153   2169   2188*  2194   2203*  2216   2237*
                    2253   2261*  2277   2285*  2301   2308*  2342   2388   2403   2449   2456*  2466   2509
                    2552   2559*  2569   2613   2656   2663*  2673   2721   2841   2849*  2859   2907   3027
                    3034*  3051   3083   3098   3131   3143*  3158   3285   3294*  3306   3322   3352*  3364*
                    3373   3402*  3414*  3423   3452*  3464*  3474   3503*  3515*  3524   3553*  3565*  3574
                    3603*  3615*  3625   3640*  3653   3686*  3692*  3697   3700   3713*  3723   3728*  3747*
                    3769*  3776*  3829*  3847*  3852   3866*  3879   3888*  3890*  3900*  3932*  3948*  3952
                    3967*  3987*  3989   3999*  4001*  4046*  4097   4163   4164   4179*  4183*  4191*  4229*
                    4258*  4295*  4305*  4359*  4377*  4456*  4494*  4519*  4581*  4602*  4609*  4662   4663
                    4712   4732*  4776*  4791*  4792   4793*  4796*  4797   4798*  4824*  4833*  4863*  4886*
                    4888*  4892*  4894*  4933*  4939*  4983*  4990*  4996*  5001*  5007*  5009*  5013*  5017*
                    5022*  5088*  5090*  5100   5107*  5153*  5303   5305*  5307*  5315*  5319*  5334   5340*
                    5345*  5391   5397*  5399*  5401*  5402*  5403*  5404   5416*  5423*  5488*  5498*
R6      =%000006     641#    643   1429*  1430*  1431
R7      =%000007     642#    644
SAVTST   001532     1105#  1563*  1564*  1862*  1863*  1904*  1905*  3732   3734   4739   4739
SCANM    026215     4457   5622#
SELECT   002632      787   1427#
SELFLG   001554     1121#  1425*  1427*  1842
SETAE    020020     3161   4400   4402   4405#
SETCON   016472     2265   2289   3152   4094#
SKPMES   026342     4777   5638#
SP      =%000006     643#    817*   859*   886*   887*   888*   889*   890*   891*   892    898*   902    903
                     904     905    906    907    912*  1433*  1436*  1437*  1443   1444   1445   1475   1478*
                    1483   1507*  1545   1578*  1603*  1607   1614*  1636*  1659   1822*  1825*  1859*  1861*
                    1869   1886*  1888   1892*  1894   1898   1900   1925   1934*  1935*  1936   1943*  1945
                    1963*  2224*  2225*  2226   2713*  2715   2718*  2720   2833*  2835   2838*  2840   2899*
                    2901   2904*  2906   3019*  3021   3024*  3026   3090*  3101   3198   3219*  3240*  3258
                    3271   3300*  3301*  3308   3309   3648*  3649*  3666   3716   3720*  3722   3771*  3774*
                    4004*  4047*  4058*  4069   4076*  4077*  4087   4088   4151*  4153   4160*  4161*  4162*
                    4186*  4188   4189   4190   4199*  4200*  4215*  4216*  4225   4226   4250*  4252   4264*
                    4280   4281   4282*  4303*  4304   4308   4310*  4311*  4313   4319   4319   4320   4321
```

```
                    4324    4325    4326*   4358    4362*   4363*   4387    4388    4407*   4412    4422*   4440    4449*
                    4450*   4451*   4452*   4453*   4454*   4455*   4468*   4476*   4503    4513    4522    4523    4524
                    4525    4526    4527    4528    4535*   4542    4586*   4587*   4588*   4589*   4612*   4613*   4617*
                    4621*   4625*   4629*   4633    4634    4640    4641    4642    4643    4663*   4669*   4672    4674
                    4675    4680*   4681*   4682    4706    4707    4711*   4712    4735*   4779*   4780*   4783*   4788*
                    4818*   4827    4848*   4851*   4865*   4870*   4874*   4896*   4904*   4907*   4911*   4914*   4919*
                    4923*   4924*   4927*   4937    4938    4954*   4955*   4958*   4959*   4968*   4969*   4975*   4977
                    4980    4985*   4993    4998*   5024    5025    5026*   5027*   5028*   5050*   5051*   5052*   5053*
                    5054*   5058*   5060    5064*   5066    5068    5076*   5077    5079    5080*   5082    5083    5084
                    5086    5104*   5131*   5132    5142*   5144    5145    5146*   5148    5150    5152    5157    5159*
                    5161*   5169*   5173    5177    5178    5182    5199*   5200*   5207    5208*   5219    5220*   5221*
                    5227    5228*   5246    5247*   5252    5253    5257*   5265    5268*   5275    5299*   5300*   5301*
                    5302*   5303*   5304*   5305    5308*   5322    5324*   5326    5336    5338    5340    5341    5342
                    5343    5344    5347*   5348*   5381*   5382    5383    5384*   5389*   5390*   5391*   5397    5423
                    5424    5425    5426*   5427*   5439    5451*   5452*   5453*   5454*   5455    5493    5494    5495
                    5496    5497*
SPRNT   020530      1689    1709    1734    3169    4550#
SPRNTA  020616      4560    4564    4569#
SPRNTB  020622      4552    4570#
SPRNTP  020554      3215    3235    3248    3265    4558#
SPRNTQ  020542      4381    4428    4434    4486    4554#
SPRNTO  020560      1702    2013    2050    2165    3191    3207    3274    4556    4559#
SPRNTI  020566      2127    4562#
SPRNT2  020604      1996    2090    2212    2249    2273    2297    2356    2364    2372    2380    2417    2425    2433
                    2441    2480    2487    2494    2501    2524    2533    2542    2584    2591    2598    2605    2629
                    2637    2646    2688    2695    2702    2709    2736    2745    2754    2761    2770    2779    2786
                    2795    2804    2811    2820    2829    2874    2881    2888    2895    2922    2931    2940    2947
                    2[illegible]    2965    2972    2981    2990    2997    3006    3015    3068    3075    3116    3123    3314
                    4567#
SPRNT3  020600      3368    3418    3469    3519    3569    3620    4566#
SR0     = 177572    736#    823     839*    1490*   1583*   1969    3819*   4342*   4350*   5459
SR1     = 177574    737#
SR2     = 177576    738#
SR3     = 172516    739#
STACK   = 001100    624#    817     858     860     1146    1433    1507    1578
START   002624      786     1425#
STARTA  002640      820     872     1426    1428#
START1  005364      869     1953    1963#   1964    3743    3788
STKLMT= 177774      629#
SWR     001136      944#    1439    1441*   1449*   1487    1580    1649    2222    2224    3148    3736    4269    4271*
                    4322    4324*   4397    4401    4420    4664    4678    4680    4688    4695    4724    4822    4830
                    4843    4846
SWREG   000176      783#    1441
SW0     = 000001    682#
SW00    = 000001    672#    682
SW01    = 000002    671#    681
SW02    = 000004    670#    680
SW03    = 000010    669#    679
SW04    = 000020    668#    678
SW05    = 000040    667#    677     4724
SW06    = 000100    666#    676     1649    3148
SW07    = 000200    665#    675     3736
SW08    = 000400    664#    674     2222
SW09    = 001000    663#    673
SW1     = 000002    681#
SW10    = 002000    662#
```

```
SW11   = 004000          661#
SW12   = 010000          660#   1487    1580
SW13   = 020000          659#
SW14   = 040000          658#
SW15   = 100000          657#
SW2    = 000004          680#
SW3    = 000010          679#
SW4    = 000020          678#
SW5    = 000040          677#   4401
SW6    = 000100          676#   4397    4420
SW7    = 000200          675#
SW8    = 000400          674#
SW9    = 001000          673#
TBITVE= 000014           715#
TCRLF  = 000200         5150    5189#
THT    = 000011         5148    5188#
TIMOUT  003516           791    1578#   1640
TKVEC  = 000060          722#
TMAP    004552          1795    1812#
TMPFAD  001562          1127#   3897    4022    4721#   4726    4728#   4750    4752#   4753#   4754    4769    4791
TMPLAD  001574          1133#   3848#   3867#   3949#   3968#   3971    4737#   4758    4760#   4761    4769    4796
TMPPT   001546          1115#   3850#   3851#   3855#   3856#   3858    3860    3862    3864    3876#   3881#   3883    3886
                        3891#   3892#   3928    3930    3950#   3951#   3956#   3957#   3959    3961    3963    3965    3988#
                        3991#   3993    4002#   4003#
TO      025241          1611    1633    4603    5529#
TPVEC  = 000064          723#
TRAPVE= 000034           721#
TRTVEC= 000014           716#
TSTMAP  001526          1102#   3834    3836    3858    3860    3870    3883    3940    3942    3959    3961    3983    3993
                        4031    4033    4040    4738#   4739#   4740#   4741#   4744    4746#   4747#   4756#   4757#   4763#
                        4764#   4771    4773
TST1    005432          1968    1970    1982#
TST10   006356          2223    2227    2236#
TST11   006446          2243    2260#
TST12   006540          2284#
TST13   006630          2307#
TST14   007304          2311    2455#
TST15   007636          2459    2558#
TST16   010174          2562    2662#
TST17   011024          2666    2848#
TST2    005564          2029#
TST20   011654          2852    3033#
TST21   012214          3037    3142#
TST22   013060          3150    3293#
TST23   013202          3351#
TST24   013300          3355    3401#
TST25   013400          3405    3451#
TST26   013504          3455    3502#
TST27   013604          3506    3552#
TST3    005666          2067#
TST30   013704          3556    3602#
TST31   014006          3606    3639#
TST32   014346          1158    1986    3643    3730#
TST4    006000          2108#
TST5    006076          2144#
TST6    006200          2187#   2229
```

F11


```
TST6A   006214          2191#   2230
TST7    006236          2202#   2229
TYPMAP  020630          1560    1832    4577#
UNEXPT  026060          4360    5604#
UP    = 000000           763#   3803    3804    3805    3806    3810
WWP   = 000004           767#   1772    1776    3177    3183
WWPBYT  012264          3154#
WWPB0   012230          3146#
WWPB1   012324          3163#   3287
WWPB2   012370          3175#   3283
WWPB3   012724          3243    3252#
WWPB4   013014          3194    3210    3278#
WWPB5   013040          3160    3166    3285#
W3X9    016560          2465    2568    2672    2858    4127#
$APTHD  001326          1046    1052#
$APTR   001510          1089#   5236    5258
$ASTAT  001342          1062#   1089    5225    5229    5256    5259    5263    5269
$ASTEN  001506          1088#
$ATYC   023572          5196    5198#
$ATY1   023546          5194#
$ATY3   023554          5138    5195#
$ATY4   023564          4839    5197#
$ATY6   024062          5256#
$ATY7   024126          5267#
$BASE   001256          1015#
$BDADR  001122           938#   1179    1182    1184    4175*   4358*
$BDDAT  001126           940#   1162    1164    1170    1173    1176    1179    1188    1431    3704*   4173*   4570*
$BELL   001172           960#   4825    4854
$CDW1   001262          1017#
$CDW2   001264          1018#
$CHARC  023542          5164*   5171    5180*   5185#
$CMTAG  001100           926#   1429
$CM3  = 000000           954#
$CM4  = 000004           954#    955#    956#    957#    958#
$CNTLU  023050          5002    5032#
$CPUOP  001230           986#
$CRLF   001177           962#   1562    3748    4834    4854    4864    4889    4895    4940    5008    5034    5094    5154
                        5191
$DBLK   024376          5310    5346    5354#
$DDW0   001266          1019#
$DDW1   001270          1020#
$DDW10  001312          1029#
$DDW11  001314          1030#
$DDW12  001316          1031#
$DDW13  001320          1032#
$DDW14  001322          1033#
$DDW15  001324          1034#
$DDW2   001272          1021#
$DDW3   001274          1022#
$DDW4   001276          1023#
$DDW5   001300          1024#
$DDW6   001302          1025#
$DDW7   001304          1026#
$DDW8   001306          1027#
$DDW9   001310          1028#
$DEVCT  001212           977#
```

```
$DEVM   001260          1016*
$DOAGN  014556          3765    3781    3787*
$DTBL   024366          5314    5350*
$ENDAD  014546           803    3783*
$ENDCT  014502          3767*
$ENDMG  014562          3770    3789*
$ENULL  014577          3777    3792*
$ENV    001222           982*   4836    5133    5203    5242
$ENVM   001223           983*   1447    5135    5140    5205
$EOP    014452          3735    3737    3758*
$EOPCT  014474          3764*   3768
$ERFLG  001103           929*   4652    4684    4686    4692*   4802    4819*   4854
$ERMAX  001115           935*   4686    4709*   4802
$ERROR  022036          1690    1703    1710    1735    1997    2014    2051    2091    2128    2166    2213    2250    2274
                        2298    2357    2365    2373    2381    2418    2426    2434    2442    2481    2488    2495    2502
                        2525    2534    2543    2585    2592    2599    2606    2629    2638    2647    2689    2696    2703
                        2710    2737    2746    2755    2762    2771    2780    2787    2796    2805    2812    2821    2830
                        2875    2882    2889    2896    2923    2932    2941    2948    2957    2966    2973    2982    2991
                        2998    3007    3016    3069    3076    3117    3124    3170    3192    3208    3216    3236    3249
                        3266    3275    3315    3369    3419    3470    3520    3570    3621    3711    4176    4369    4382
                        4429    4435    4487    4817*   5440
$ERRPC  001116           936*   1162    1164    1166    1168    1170    1173    1179    1182    1184    1186    4827*   4828*
                        4829    4854    4870    4877
$ERRTB  002314          1156    1291*
$ERRTY  022230          4832    4862*
$ERTTL  001112           933*   4483*   4826*   4854
$ESCAP  001170           959*   4708*   4849    4851    4854
$ETABL  001222           981*
$ETEND  001326          1035*   1058
$FATAL  001204           974*   5246*
$FFLG   024152          5194*   5197*   5240    5249*   5279*
$FILLC  001154           952*   5157    5191
$FILLS  001153           951*   5191
$GDADR  001120           937*   1162    1164    1166    1168    1170    1173    1176    1179    1188    3703*   4174*   4550*
                        4559*   4562*   4563*   4567*   4568*
$GDDAT  001124           939*   1162    1164    1166    1170    1173    1176    1179    3707*   3709*   4172*   4551*   4569*
$GET42  014536          3778*
$HD     =  000000        603
$HIBTS  001326          1053*
$HIOCT  023236          1866    1879    1890    1893*   1895    1906    5081*   5093*
$ICNT   001104           930*   4699*   4700    4702*   4801
$ILLUP  000740           884     915*
$ITEMB  001114           934*   4829*   4838    4854    4867
$LF     001200           963*   4854    5023    5034    5094    5191
$LFLG   024151          5225    5250*   5278*
$LPADR  001106           931*   1964*   1965*   2229*   4690*   4706*   4711    4789*   4801
$LPERR  001110           932*   4690    4707*   4801    4848
$MADR1  001234           999*
$MADR2  001240          1003*
$MADR3  001244          1006*
$MADR4  001250          1009*
$MAIL   001202           972*   1054    1058    1446    1988    2032    2070    2111    2147    2190    2205    2239    2263
                        2287    2313    2461    2564    2668    2854    3039    3145    3296    3357    3407    3457    3508
                        3558    3608    3645    4705    4836    5133
$MAMS1  001232           993*
$MAMS2  001236          1001*
```

```
$MAMS3  001242          1004#
$MAMS4  001246          1007#
$MBADR  001330          1054#
$MFLG   024150          5195*   5201    5251*   5277#
$MSGAD  001216           979#   5211*   5214    5263*
$MSGLG  001220           980#   5216*   5262*
$MSGTY  001202           973#   5209    5217*   5244    5248*   5260    5264*
$MTYP1  001233           994#
$MTYP2  001237          1002#
$MTYP3  001243          1005#
$MTYP4  001247          1008#
$MXCNT  022034          4703    4801#
$NULL   001152           950#   5159    5191
$NWTST= 000001          1973#   1975    2020#   2022    2058#   2060    2099#   2101    2135#   2137    2177#   2179    2180
                        2198#   2200    2232#   2234    2257#   2281#   2304#   2452#   2555#   2659#   2845#   3030#   3134#
                        3136    3290#   3326#   3328    3376#   3378    3426#   3428    3477#   3479    3527#   3529    3577#
                        3579    3628#   3630
$OCNT   024632          5388*   5418*   5431#
$OMODE  024634          5383*   5387*   5392    5395*   5406*   5433#
$OVER   021320          4665    4683    4691    4701    4710#
$PASS   001210           976#   1446*   3718    3761*   3762*   3771    3789    4697    4716    4742    4802
$PASTM  001334          1056#
$PRINT  023240           910    1455    1493    1516    1561    1569    1587    1600    1610    1632    1669    1813    1819
                        1827    1854    1881    1926    1938    3747    3769    3776    4183    4359    4377    4456    4494
                        4519    4581    4602    4609    4732    4776    4824    4833    4863    4886    4888    4892    4894
                        4933    4939    4983    4990    4996    5001    5007    5009    5013    5017    5022    5088    5090
                        5100#   5153    5345    5416    5488    5498
$PRIOR  001254          1012#
$PWRAD  000734           913#
$PWRDN  000610           884#    908    1434
$PWRMG  000730           911#
$PWRUP  000656           893     898#
$QUES   001176           961#   4854    5014    5034    5091    5094    5191
$RDCHR  022530          4954#   4976
$RDLIN  022564          4968#   5059
$RDOCT  023066          1860    1887    1944    5050#
$RDSZ = 000010          4961#
$SAVR6  000744           892*    898     899*    900*    917#
$SCOPE  021046          1983    2030    2068    2109    2145    2188    2203    2237    2261    2285    2308    2456    2559
                        2663    2849    3034    3143    3294    3352    3402    3452    3503    3553    3603    3640    3728
                        4661#
$SETUP= 000010           774#   1434    1436    1455    3760    4846
$STUP = 177777           774#
$SVLAD  021264          4673    4704#
$SVPC = 000010           801#    806
$SWR  = 167400           592#    603     609     610     611     612     613     614     615     616     914     958     959
                         960    1988    2032    2070    2111    2147    2190    2205    2239    2263    2287    2313    2461
                        2564    2668    2854    3039    3145    3296    3357    3407    3457    3508    3558    3608    3645
                        3755    3760    3779    3788    3789    4653    4654    4655    4656    4657    4664    4676    4678
                        4679    4684    4685    4686    4693    4694    4695    4707    4710    4801    4809    4810    4811
                        4812    4813    4822    4830    4843    4846    4854
$SWREG  001224           984#   1449
$SWRMK= 000340           592#    616     617    4657    4658    4680    4681
$TESTN  001206           975#   1988*   2032*   2070*   2111*   2147*   2190*   2205*   2239*   2263*   2287*   2313*   2461*
                        2564*   2668*   2854*   3039*   3145*   3296*   3357*   3407*   3457*   3508*   3558*   3608*   3645*
                        4705*
```

```
$TIMES  001166          867*   958*  3730*  3760*  4693*  4700   4703*  4801
$TKB    001144          947*  4948   4958
$TKS    001142          946*  4948   4956
$TMPO   001156          954*  1168   1170   1173   1176   1184   1186   1188   3697*  3698*  3699*  3705   4055*
                       4065*  4067*  4554*  4558*  4566*
$TMP1   001160          955*  1176   1184   1186   1188   3700*  3701*  4555*
$TMP2   001162          956*  1176   1525*  1526*  1529*  1540*  1844   1892   3702*
$TMP3   001164          957*  1176   1190   1515*  1531*  1537*  1539*  1541*  1893   3696*  5439*
$TN   = 000032          592*   603   1973   1988*  2020   2032*  2058   2070*  2099   2111*  2135   2147*  2177
                       2190*  2198   2200   2205*  2219   2232   2239*  2243   2257   2263*  2281   2287*  2304
                       2307   2313*  2452   2455   2461*  2555   2558   2564*  2659   2662   2668*  2845   2848
                       2854*  3030   3033   3039*  3134   3145*  3290   3296*  3326   3351   3357*  3376   3401
                       3407*  3426   3451   3457*  3477   3502   3508*  3527   3552   3558*  3577   3602   3608*
                       3628   3639   3645*
$TPB    001150          949*  5177*  5191
$TPFLG  001155          953*  5127   5191
$TPS    001146          948*  5175   5191
$TSTM   001332         1055*
$TSTNM  001102          868*   928*  2228*  3731*  4652   4682   4704*  4705   4710   4718   4780   4802   4821
                       4854
$TTYIN  023055         4970   4971   4987   5010   5028   5033*
$TYPAD  024674         1604   1615   1637   4187   4920   5450*
$TYPDS  024154         3775   4915   5298*
$TYPE   023266         5105   5127*  5222
$TYPEC  023476         5156   5163   5170   5175*  5176
$TYPEX  023544         5181   5183   5186*
$TYPOC  024432         1826   4875   4908   5386*
$TYPON  024446         5385   5388*
$TYPOS  024406         4622   4630   4784   4928   5381*
$UNIT   001214          978*
$UNITM  001336         1057*
$USWR   001226          985*
$VECT1  001252         1010*
$VECT2  001253         1011*
$VERPC  001512         1094*  1164   1166   1168   1170   1173   1179   1182   1184   1186   4877*  4878*
$XTSTR  021062         4667*
$OFILL  024633         5382*  5386*  5396   5432*
$40CAT= ****** U       4664   4832
.     = 032110          777*   781*   793*   801    802*   804*   806*   807*   895    916    925*   964    1042
                       1043*  1045*  1047*  1124*  1273*  1432   1459*  1461   3694   3789   3793   4801   4802
                       4854   4948   5033*  5034*  5094   5191   5280*  5354*  5863*
.BGERR  001622         1151*  3667
.BGEXI  001624         1152*  3668
.BRGOB  001620         1150*  3650
.CONST  001606         1138*  1945*  2191   2206
.EIGHT  001636         1157*  5331
.ERRTB  001634         1156*  4883
.MPPAT  001630         1154*  2240
.MPRX   001616         1149*  3176   3182   3205   3242   4364   4408   4423   4478   4536
.MPRO   001614         1148*  3199
.PBTRP  001626         1153*  3186
.PESRV  001632         1155*  3190   3197   4405
.RESRV  001612         1147*
.STACK  001610         1146*  1963
.TST32  001640         1158*  4735
.$ASTA= ****** U       5195   5198
```

DZQMCB.P11 CROSS REFERENCE TABLE -- USER SYMBOLS

```
.$X    = 001326          1042#   1047
.3X9     007324          2462#
.9X13    011674          3040#
```

```
ABORT    592*   848   3842  3875  3913  4085  4178  4197  4255  4290  4339
CKWD     592*  1700   1994  2011  2088  2125  2163  2210  2247  2271  2295  2354  2362  2370  2378
        2415  2423   2431  2439  2478  2485  2492  2499  2522  2531  2540  2582  2589  2596  2603
        2626  2635   2644  2686  2693  2700  2707  2734  2743  2752  2759  2768  2777  2784  2793
        2802  2809   2818  2827  2872  2879  2886  2893  2920  2929  2938  2945  2954  2963  2970
        2979  2988   2995  3004  3013  3066  3073  3114  3121  3233  3263  3312  3366  3416  3467
        3517  3567   3618
CKWD2    592*  1993   2086  2208  2246  2352  2361  2369  2377  2413  2422  2430  2438  2476  2484
        2491  2498   2520  2530  2539  2580  2588  2595  2602  2624  2634  2643  2684  2692  2699
        2706  2732   2742  2751  2758  2767  2776  2783  2792  2801  2808  2817  2826  2870  2878
        2885  2892   2918  2928  2937  2944  2953  2962  2969  2978  2987  2994  3003  3012  3064
        3072  3112   3120  3311
COMMEN     1*   592*   725*  1419  1957
ENDCOM     1*   592*   603   725*  1423  1961
ERROR    625*
ESCAPE     1*   725*
GETPRI     1*
LDPDR    592*  3802   3804  3805  3806  3810
MORETA   918*  1037
MULT       1*   725*
NEWTST     1*   592*   725*  1973  2020  2058  2099  2135  2177  2198  2232  2257  2281  2304  2452
        2555  2659   2845  3030  3134  3290  3326  3376  3426  3477  3527  3577  3628
POP        1*   592*   725*   902  3258  3271  4069  4086  4188  4280  4318  4386  4412  4440  4503
        4513  4521   4541  4633  4640  5082  5252  5253  5265  5274  5340  5493
PRINT    592*  1493   1516  1561  1569  1586  1600  1610  1632  1669  1813  1819  1827  1854  1880
        1926  1937   3747  4183  4359  4377  4456  4494  4519  4581  4732  4775  5488  5498
PUSH       1*   592*   725*   886  3218  3240  4058  4076  4159  4198  4215  4311  4362  4407  4422
        4448  4468   4475  4534  4585  4611  5052  5198  5200  5221  5256  5267  5298  5450
RDCHR    592*  4973
RDDEC    592*
RDLIN    592*  5055
RDOCT    592*  1857   1884  1941
REPORT     1*   592*
RESREG   592*
SAVREG   592*
SCOPE    626*
SCOPEX  4646*  4712
SCOPIN  4646*  4662
SETPRI     1*
SETUP      1*   592*   725*  1428
SIMTRP   592*  1823   1857  1884  1941  3772  4619  4627  4781  4872  4905  4912  4925  4973  5056
        5102
SKIP       1*   725*  2243
SLASH      1*   592*   725*  1213  1223
SPACE    725*
STARS      1*   592*   725*   797   808   813   879   918   964   965  1037  1041  1048  1059  1091
        1093  1140   1144  1159  1161  1194  1196  1497  1506  1574  1577  1643  1646  1676  1680
        1717  1720   1741  1750  1761  1765  1807  1810  1847  1849  1973  1981  2020  2028  2058
        2066  2099   2107  2135  2143  2173  2176  2177  2186  2198  2201  2232  2235  2257  2259
        2281  2283   2304  2306  2346  2348  2392  2394  2407  2409  2452  2454  2470  2472  2513
        2515  2555   2557  2574  2576  2617  2619  2659  2661  2677  2679  2725  2727  2845  2847
        2863  2865   2911  2913  3030  3032  3055  3057  3087  3089  3103  3105  3134  3141  3290
        3292  3326   3350  3376  3400  3426  3450  3477  3501  3527  3551  3577  3601  3628  3638
        3749  3795   3801  3823  3825  3922  3927  4012  4017  4050  4053  4072  4074  4091  4093
        4101  4103   4124  4126  4156  4158  4192  4194  4285  4287  4332  4336  4353  4357  4391
        4394  4415   4417  4443  4447  4531  4533  4546  4549  4573  4576  4646  4802  4854  4944
```

```
        4961    5035    5095    5099    5109    5191    5285    5355    5505    5508    5644    5646    5789    5791    5843
        5845
SWRSU      1#  1436#
SWROR3   592#  2713    2718    2833    2838    2899    2904    3019    3024    4151
S1REGS  1973#  1975    2022    2060    2101    2137    2180
TYPADR   592#  1603    1614    1636    4186    4919
TYPBIN     1#   725#
TYPBN    592#
TYPBYT   592#  4779    4922
TYPDEC     1#   725#  3771    4911
TYPDS    592#  3772    4912
TYPE     592#   910    1455    1493    1516    1561    1569    1587    1600    1610    1632    1669    1813    1819    1827
        1854   1881    1926    1938    3747    3769    3776    4183    4359    4377    4456    4494    4519    4581    4602
        4609   4732    4776    4824    4833    4863    4886    4888    4892    4894    4933    4939    4983    4990    4996
        5001   5007    5009    5012    5017    5022    5088    5090    5153    5345    5416    5488    5498
TYPNAM     1#   592#  1453
TYPNUM     1#   725#
TYPOC    592#  1823    4872    4905
TYPOCS     1#   725#  4617    4625
TYPOCT     1#   725#  1822    4870    4904
TYPON    592#
TYPOS    592#  4619    4627    4781    4925
TYPTXT     1#   725#
$CKWD    592#  1701    1733    1995    2012    2049    2089    2126    2164    2211    2248    2272    2296    2355    2363
        2371   2379    2416    2424    2432    2440    2479    2486    2493    2500    2523    2532    2541    2583    2590
        2597   2604    2627    2636    2645    2687    2694    2701    2708    2735    2744    2753    2760    2769    2778
        2785   2794    2803    2810    2819    2828    2873    2880    2887    2894    2921    2930    2939    2946    2955
        2964   2971    2980    2989    2996    3005    3014    3067    3074    3115    3122    3168    3234    3264    3273
        3313   3367    3417    3468    3518    3568    3619
$INDN    592#  2007    2044    2072    2149    2159
$INMM    592#  1990    2034    2083    2113    2122    2192    2207    2244    2266    2290    2316    2350    2395    2411
        2464   2474    2517    2567    2578    2621    2671    2682    2730    2857    2868    2916    3041    3059    3092
        3107   3153    3299    3361    3411    3461    3512    3562    3612    3646    4094
$MMDN    592#  2017    2054    2077    2153    2169
$MMUP    592#  2001    2038    2095    2116    2131    2194    2216    2253    2277    2301    2342    2388    2403    2449
        2466   2509    2552    2569    2613    2656    2673    2721    2841    2859    2907    3027    3051    3083    3098
        3131   3321    3373    3423    3474    3524    3574    3625    4097
$$CKWD   592#  1689    1702    1709    1734    1996    2013    2050    2090    2127    2165    2212    2249    2273    2297
        2356   2364    2372    2380    2417    2425    2433    2441    2480    2487    2494    2501    2524    2533    2542
        2584   2591    2598    2605    2628    2637    2646    2688    2695    2702    2709    2736    2745    2754    2761
        2770   2779    2786    2795    2804    2811    2820    2829    2874    2881    2888    2895    2922    2931    2940
        2947   2956    2965    2972    2981    2990    2997    3006    3015    3068    3075    3116    3123    3169    3191
        3207   3215    3235    3248    3265    3274    3314    3368    3418    3469    3519    3569    3620    4390    4429
        4433   4485
$$CMRE   918#
$$CMTM   918#   954     955     956     957
$$ESCA     1#   725#
$$NEWT     1#   725#  1973    2020    2058    2099    2135    2177    2198    2232    2257    2281    2304    2452    2555
        2659   2845    3030    3134    3290    3326    3376    3426    3477    3527    3577    3628
$$SKIP     1#   725#  2243
.BGHD    3628#  3630
.C11     3134#  3136
.DIDBH   3426#  3428
.DIDBL   3376#  3378
.DIDO    3326#  3328
.DIPDO   3477#  3479
```

```
.DPDBH   3577#   3579
.DPDBL   3527#   3529
.EQUAT      1#    592#    620
.ERROR    592#   1690   1703   1710   1735   1997   2014   2051   2091   2128   2166   2213   2250   2274   2298
          2357   2365   2373   2381   2418   2426   2434   2442   2481   2488   2495   2502   2525   2534   2543
          2585   2592   2599   2606   2629   2638   2647   2689   2696   2703   2710   2737   2746   2755   2762
          2771   2780   2787   2796   2805   2812   2821   2830   2875   2882   2889   2896   2923   2932   2941
          2948   2957   2966   2973   2982   2991   2998   3007   3016   3069   3076   3117   3124   3170   3192
          3208   3216   3236   3249   3266   3275   3315   3369   3419   3470   3520   3570   3621   3710   4176
          4369   4382   4429   4435   4487   5440
.HEADE      1#    592#    593
.KT11       1#    592#    727
.SCOPE    592#   1982   2029   2067   2108   2144   2187   2202   2236   2260   2284   2307   2455   2558   2662
          2848   3033   3142   3293   3351   3401   3451   3502   3552   3602   3639   3727
.SETUP      1#    592#    774
.SWRHI      1#    592#    604
.SWRLO    592#    617#
.TM7     2198#   2200
.$ACT1      1#    592#    797
.$APTB      1#    592#    965#
.$APTH      1#    592#   1037
.$APTY      1#    592#   5191
.$ASTA      1#    592#   1059
.$CATC      1#    592#    774
.$CMTA      1#    592#    918
.$DB2D      1#
.$DB2O      1#
.$DIV       1#
.$EOP       1#    592#   3749
.$ERRO      1#    592#   4802
.$ERRT      1#    592#   4854
.$MULT      1#
.$POWE      1#    592#    879
.$RAND      1#
.$RDDE      1#
.$RDOC      1#    592#   5035
.$READ      1#    592#   4944
.$R2AZ      1#
.$SAVE      1#
.$SB2D      1#
.$SB2O      1#
.$SCOP      1#    592#   4646
.$SIZE      1#
.$SUPR      1#
.$TRAP      1#
.$TYPB      1#
.$TYPD      1#    592#   5285
.$TYPE      1#    592#   5109
.$TYPO      1#    592#   5355
.$40CA      1#
```

```
ADC     1541  1865  1912  4608  4636  5476
ADCB    3679
ADD      846   851   858   860  1540  1550  1628  1786  1792  1803  1833  1864  1911  1935  1949
        1950  1965  2000  2075  3165  3204  3228  3241  3674  3699  3838  3854  3872  3911  3936
        3955  3980  4068  4262  4264  4266  4267  4268  4271  4272  4273  4274  4282  4505  4503
        4607  4635  4788  4789  4818  4883  4899  4901  5077  5101  5146  5208  5220  5228  5247
        5311  5319  5384  5394  5468  5475  5486
ASL     1532  1533  1534  1535  1536  1538  1552  1712  1793  1871  1913  3223  3224  3839  3855
        3937  3956  4060  4061  4062  4063  4064  4066  4251  4510  4614  4637  4793  4798  4880
        4881  4882  5070  5072  5074  5471  5477  5481
ASLB    1557  1804  3873  3881  3981  3991  4515  5324
ASR      843   852  4038  4080  4241  4246  4720  5215
ASRB    3975
BCC      854  1713  2294  4639  5325
BCS      845  2270  3689  3977  4030  4039  4082  4209  4220  4242  4247
BEQ      822   863  1448  1462  1528  1549  1568  1609  1619  1623  1625  1631  1688  1701  1708
        1733  1757  1791  1878  1901  1933  1947  1968  1995  2012  2049  2089  2126  2164  2211
        2223  2243  2248  2272  2296  2355  2363  2371  2379  2416  2424  2432  2440  2479  2486
        2493  2500  2523  2532  2541  2583  2590  2597  2604  2627  2636  2645  2687  2694  2701
        2708  2735  2744  2753  2760  2769  2778  2785  2794  2803  2810  2819  2828  2873  2880
        2887  2894  2921  2930  2939  2946  2955  2964  2971  2980  2989  2996  3005  3014  3067
        3074  3115  3122  3147  3149  3168  3226  3234  3264  3273  3286  3313  3367  3417  3468
        3518  3568  3619  3654  3660  3706  3719  3735  3781  3831  3846  3853  3861  3865  3880
        3884  3887  3896  3903  3917  3934  3943  3947  3953  3962  3966  3976  3984  3986  3990
        3994  3998  4008  4021  4026  4034  4041  4045  4057  4171  419F  4202  4213  4224  4244
        4249  4254  4270  4276  4289  4292  4315  4323  4341  4349  4374  4396  4400  4419  4467
        4474  4507  4538  4606  4679  4683  4685  4689  4698  4713  4717  4723  4725  4745  4749
        4751  4759  4768  4820  4823  4847  4850  4885  4891  4898  4932  4994  5065  5136  5149
        5202  5206  5226  5233  5237  5241  5243  5411  5458  5460
BGE     4701  5183
BGT     3765  5067  5419
BHI     1868  1870  1896  1899  1908  1910  3243  4687  4731  5333
BHIS    4719  4727
BIC     1529  1566  1653  1687  1697  1707  1731  1776  1861  1934  1948  1951  2225  3183  3221
        3227  3230  3231  3260  3262  3298  3762  3848  3949  4681  4740  4741  4746  4756  4757
        4760  4763  4764  4792  4797  4959  5076  5408  5467  5474
BIS     1523  1524  1546  1660  1782  1783  1784  1785  1904  1905  3158  3177  3229  3723  3867
        3891  3892  3968  4002  4003  4225  4226  4252  4409  4729  4752  5327  5328  5413  5414
BISB    1597  4867
BIT      823  1487  1521  1580  1594  1620  1649  1692  1770  1796  1798  1801  1817  1875  1977
        1900  1916  1918  1921  1923  1952  1969  2001  2017  2038  2054  2077  2095  2116  2131
        2153  2169  2194  2216  2222  2253  2277  2301  2342  2388  2403  2449  2466  2509  2552
        2569  2613  2656  2673  2721  2841  2859  2907  3027  3051  3083  3098  3131  3148  3154
        3156  3200  3211  3213  3244  3285  3303  3306  3318  3322  3373  3423  3474  3524  3574
        3625  3653  3732  3734  3736  3834  3836  3843  3845  3852  3858  3860  3862  3864  3870
        3877  3879  3883  3886  3893  3895  3914  3916  3928  3930  3940  3942  3944  3946  3952
        3959  3961  3963  3965  3983  3985  3989  3993  3995  3997  4018  4020  4031  4033  4040
        4042  4044  4097  4210  4212  4221  4223  4227  4243  4248  4253  4288  4397  4401  4420
        4426  4471  4473  4501  4594  4596  4664  4678  4688  4695  4724  4744  4750  4754  4758
        4761  4765  4767  4822  4830  4846  5459
BITB    1447  5135  5140  5171  5205
BLO     1664  1788  1835  1867  1897  1907  3206  4404  4988  5332
BLOS    4770  4972
BLT     5069  5162  5316  5420
BMI     1795  1915  3203  3680  3691  3857  3882  3939  3958  3982  3992  4366  4910  5231  5271
        5323
```

```
BNE      819  824  829  866  901 1432 1454 1470 1482 1488 1522 1581 1595 1599 1621
        1650 1668 1693 1715 1725 1739 1771 1797 1799 1802 1818 1841 1843 1876 1889
        1891 1917 1919 1922 1924 1953 1970 2002 2018 2039 2055 2078 2096 2117 2132
        2154 2170 2195 2217 2227 2254 2278 2302 2341 2343 2386 2389 2402 2404 2447
        2450 2467 2507 2510 2547 2550 2553 2570 2611 2614 2651 2654 2657 2674 2717
        2722 2837 2842 2860 2903 2908 3023 3028 3048 3050 3052 3090 3082 3084 3097
        3099 3128 3130 3132 3155 3157 3164 3179 3185 3201 3212 3214 3245 3304 3307
        3319 3323 3374 3424 3475 3525 3575 3626 3656 3664 3721 3733 3737 3739 3741
        3835 3837 3844 3859 3863 3871 3878 3894 3915 3929 3931 3941 3945 3960 3964
        3972 4019 4023 4032 4043 4096 4150 4166 4182 4211 4222 4228 4278 4317 4338
        4347 4368 4398 4402 4411 4421 4427 4439 4472 4482 4499 4502 4518 4578 4580
        4595 4597 4599 4665 4696 4743 4755 4762 4766 4772 4774 4794 4799 4831 4837
        4868 4903 4918 4979 4981 5000 5005 5020 5134 5141 5143 5151 5158 5172 5179
        5204 5210 5213 5245 5261 5321 5409 5437 5484 5492
BPL      847 1554 1558 1780 1805 1873 3247 3282 3841 3874 4084 4376 4425 4480 4493
        4512 4516 4844 4957 5128 5176 5306 5337 5407
BR       615  856  895  916 1426 1440 1457 1475 1542 1555 1605 1629 1640 1657 1800
        1874 1920 1929 2230 2256 3194 3210 3251 3658 3682 3708 3868 3889 3898 3907
        3969 3973 4000 4024 4036 4180 4236 4301 4372 4385 4431 4504 4514 4540 4552
        4556 4560 4564 4584 4604 4667 4673 4676 4691 4694 4787 4842 4876 4909 4916
        4921 4935 4992 5003 5011 5015 5078 5092 5130 5155 5165 5174 5181 5196 5218
        5235 5273 5318 5335 5385 5400 5422 5442 5480
BVC     3690
BVS     3685
CLC     2267
CLR      814  831  840  867  899 1430 1446 1451 1452 1465 1466 1486 1490 1510 1511
        1512 1514 1515 1531 1579 1583 1590 1591 1661 1662 1665 1672 1685 1696 1730
        1754 1755 1767 1768 1839 1851 1852 1853 1862 1863 1954 1955 2288 2315 2349
        2462 2473 2516 2566 3040 3058 3151 3239 3662 3677 3730 3760 3807 3808 3809
        3811 3813 3814 3815 3816 3817 3827 3828 3832 3901 3905 4055 4075 4204 4239
        4293 4298 4327 4329 4342 4406 4460 4461 4462 4465 4469 4490 4497 4539 4551
        4591 4693 4708 4747 4779 4866 4923 4969 4998 5062 5063 5272 5313 5398 5445
        5456 5490
CLRB     868 1425 3173 3731 3978 4692 5006 5021 5087 5180 5249 5250 5251 5339
CLV     3687
CMP      862 1431 1443 1461 1476 1545 1567 1607 1624 1659 1663 1700 1787 1834 1866
        1869 1895 1898 1906 1909 1994 2011 2088 2125 2163 2210 2226 2247 2271 2295
        2354 2362 2370 2378 2415 2423 2431 2439 2478 2485 2492 2499 2522 2531 2540
        2582 2589 2596 2603 2626 2635 2644 2686 2693 2700 2707 2734 2743 2752 2759
        2768 2777 2784 2793 2802 2809 2818 2827 2872 2879 2886 2893 2920 2929 2938
        2945 2954 2963 2970 2979 2988 2995 3004 3013 3066 3073 3114 3121 3163 3198
        3205 3233 3242 3263 3312 3366 3416 3467 3517 3567 3618 3705 3738 3971 4022
        4170 4195 4269 4277 4316 4322 4403 4674 4700 4712 4726 4730 4769 4971 4997
        5232 5236 5331
CMPB    2048 3272 4682 4686 4718 4836 4917 4978 4999 5004 5019 5066 5068 5133 5148
        5160 5157 5178 5182 5203
COM     1617 2074 2085 2151 2161 2317 2318 2360 2368 2376 2384 2387 2397 2398 2399
        2400 2421 2429 2437 2445 2448 2505 2508 2528 2529 2537 2538 2548 2551 2609
        2612 2632 2633 2641 2642 2652 2655 2740 2741 2749 2750 2765 2766 2774 2775
        2790 2791 2799 2800 2815 2816 2824 2825 2926 2927 2935 2936 2951 2952 2960
        2961 2976 2977 2985 2986 3001 3002 3010 3011 3043 3045 3061 3063 3094 3095
        3109 3111
COMB    3281
DEC      828 1481 1526 1613 1635 1840 2046 2340 2385 2401 2446 2506 2546 2549 2610
        2650 2653 2716 2836 2902 3022 3047 3049 3079 3081 3096 3127 3129 3465 3616
        3655 3659 3720 3763 4149 4167 4181 4346 4483 4563 4879 4986 5483 5491
```

DZQMCB.P11 CROSS REFERENCE TABLE -- PERMANENT SYMBOLS

```
DECB    5161  5164  5406  5418
EMT      625
HALT     781   848   871   894   915  1572  1638  3842  3875  3913  4085  4178  4197  4256  4290
        4339  4845  5129  5443
INC      900  1453  1492  1547  1585  1616  1618  2037  3159  3284  3663  3676  3724  3761  4350
        4484  4500  4699  4753  4826  5239  5248  5317  5412  5421  5438
INCB    3681  3688  4083  4704  4819  5184
IOT      626
JMP      786   787   789   790   791   820   850   861   869   872  1463  1673  1845  1986  2311
        2459  2562  2666  2852  3037  3150  3160  3166  3283  3287  3355  3405  3455  3506  3556
        3606  3643  3669  3693  3743  3788  4714
JSR      864   870   910  1455  1491  1493  1516  1560  1561  1569  1584  1587  1600  1604  1610
        1615  1632  1637  1669  1689  1690  1702  1703  1709  1710  1734  1735  1752  1759  1812
        1813  1819  1826  1827  1832  1854  1860  1881  1887  1926  1938  1944  1966  1971  1983
        1990  1991  1996  1997  2003  2007  2008  2013  2014  2019  2030  2034  2035  2040  2044
        2045  2050  2051  2056  2068  2072  2073  2079  2083  2084  2090  2091  2097  2109  2113
        2114  2118  2122  2123  2127  2128  2133  2145  2149  2150  2155  2159  2160  2165  2166
        2171  2188  2192  2196  2203  2207  2212  2213  2218  2237  2241  2244  2249  2250  2255
        2261  2265  2266  2268  2273  2274  2279  2285  2289  2290  2292  2297  2298  2303  2308
        2316  2344  2350  2356  2357  2364  2365  2372  2373  2380  2381  2390  2395  2405  2411
        2417  2418  2425  2426  2433  2434  2441  2442  2451  2456  2464  2465  2468  2474  2480
        2481  2487  2488  2494  2495  2501  2502  2511  2517  2524  2525  2533  2534  2542  2543
        2554  2559  2567  2568  2571  2578  2584  2585  2591  2592  2598  2599  2605  2606  2615
        2621  2628  2629  2637  2638  2646  2647  2658  2663  2671  2672  2675  2682  2688  2689
        2695  2696  2702  2703  2709  2710  2723  2730  2736  2737  2745  2746  2754  2755  2761
        2762  2770  2771  2779  2780  2786  2787  2795  2796  2804  2805  2811  2812  2820  2821
        2829  2830  2843  2849  2857  2858  2861  2868  2874  2875  2881  2882  2888  2889  2895
        2896  2909  2916  2922  2923  2931  2932  2940  2941  2947  2948  2956  2957  2965  2966
        2972  2973  2981  2982  2990  2991  2997  2998  3006  3007  3015  3016  3029  3034  3041
        3053  3059  3068  3069  3075  3076  3085  3092  3100  3107  3116  3117  3123  3124  3133
        3143  3152  3153  3161  3162  3169  3170  3191  3192  3207  3208  3215  3216  3235  3236
        3248  3249  3265  3266  3274  3275  3288  3289  3294  3299  3314  3315  3324  3352  3361
        3364  3368  3369  3375  3402  3411  3414  3418  3419  3425  3452  3461  3464  3469  3470
        3476  3503  3512  3515  3519  3520  3526  3553  3562  3565  3569  3570  3576  3603  3612
        3615  3620  3621  3627  3640  3647  3686  3692  3711  3725  3728  3742  3745  3746  3747
        3769  3775  3776  3783  3912  4009  4095  4099  4176  4183  4187  4229  4258  4295  4305
        4359  4369  4377  4381  4382  4384  4428  4429  4432  4434  4435  4437  4456  4463  4486
        4487  4494  4519  4581  4602  4609  4622  4630  4732  4776  4784  4824  4832  4833  4839
        4863  4875  4886  4888  4892  4894  4908  4915  4920  4928  4933  4939  4976  4983  4990
        4996  5001  5007  5009  5013  5017  5022  5059  5088  5090  5105  5138  5153  5156  5163
        5170  5222  5345  5416  5440  5488  5498
MOV      816   817   825   826   827   830   832   833   834   835   836   837   838   839   841
         842   849   855   857   859   884   885   886   887   888   889   890   891   892   893
         898   902   903   904   905   906   907   908   909   912  1429  1433  1434  1435  1436
        1437  1438  1441  1442  1444  1445  1449  1460  1464  1471  1472  1473  1474  1477  1478
        1479  1480  1483  1489  1507  1508  1509  1513  1519  1520  1525  1530  1551  1559  1563
        1564  1565  1578  1582  1592  1596  1603  1614  1626  1627  1636  1648  1651  1652  1656
        1666  1682  1683  1684  1686  1694  1695  1706  1722  1723  1727  1728  1729  1753  1758
        1769  1772  1774  1789  1816  1822  1825  1830  1837  1838  1844  1850  1859  1879  1886
        1892  1893  1894  1902  1903  1930  1931  1936  1943  1945  1963  1964  1988  1992  1993
        2010  2032  2070  2076  2087  2111  2147  2190  2191  2193  2205  2206  2209  2224  2239
        2240  2242  2245  2246  2263  2264  2269  2287  2293  2313  2314  2319  2320  2321  2322
        2323  2325  2326  2327  2328  2330  2331  2332  2333  2335  2336  2337  2338  2351  2353
        2361  2369  2377  2396  2410  2412  2414  2422  2430  2438  2461  2463  2475  2477  2484
        2491  2498  2518  2519  2521  2530  2539  2564  2565  2577  2579  2581  2588  2595  2602
        2620  2622  2623  2625  2634  2643  2668  2669  2670  2680  2681  2683  2685  2692  2699
```

```
        2706  2713  2714  2715  2718  2719  2720  2728  2729  2731  2733  2742  2751  2758  2767
        2776  2783  2792  2801  2808  2817  2926  2833  2834  2835  2838  2839  2840  2854  2855
        2856  2866  2867  2869  2871  2878  2885  2892  2899  2900  2901  2904  2905  2906  2914
        2915  2917  2919  2928  2937  2944  2953  2962  2969  2978  2987  2994  3003  3012  3019
        3020  3021  3024  3025  3026  3039  3042  3044  3046  3060  3062  3065  3072  3090  3091
        3093  3101  3106  3108  3110  3113  3120  3145  3176  3182  3186  3190  3197  3199  3202
        3209  3220  3240  3246  3258  3259  3269  3270  3271  3280  3296  3297  3300  3301  3302
        3308  3309  3310  3311  3357  3358  3359  3360  3362  3363  3365  3372  3407  3408  3409
        3410  3412  3413  3415  3422  3457  3458  3459  3460  3462  3463  3466  3473  3508  3509
        3510  3511  3513  3514  3516  3523  3558  3559  3560  3561  3563  3564  3566  3573  3608
        3609  3610  3611  3613  3614  3617  3624  3645  3648  3649  3650  3651  3652  3665  3666
        3667  3668  3673  3675  3696  3697  3700  3702  3703  3704  3707  3709  3716  3766  3771
        3774  3780  3803  3804  3805  3806  3810  3812  3818  3819  3826  3829  3833  3847  3849
        3850  3851  3866  3876  3888  3890  3897  3900  3904  3906  3909  3910  3918  3919  3932
        3935  3948  3950  3951  3954  3967  3987  3988  3999  4001  4004  4035  4046  4047  4054
        4058  4059  4069  4076  4077  4078  4079  4087  4088  4096  4127  4129  4130  4131  4132
        4134  4135  4136  4137  4139  4140  4141  4142  4144  4145  4146  4147  4151  4152  4153
        4160  4161  4162  4163  4164  4165  4166  4169  4172  4173  4174  4175  4186  4188  4189
        4190  4199  4200  4203  4205  4214  4215  4216  4232  4233  4235  4239  4250  4257  4261
        4265  4279  4280  4281  4294  4299  4303  4304  4311  4312  4318  4328  4337  4343  4344
        4345  4358  4362  4363  4364  4387  4388  4405  4407  4408  4412  4422  4423  4440  4449
        4450  4451  4452  4453  4454  4455  4459  4464  4468  4470  4476  4478  4503  4509  4513
        4522  4523  4524  4525  4526  4527  4528  4535  4536  4542  4550  4554  4555  4558  4559
        4562  4566  4567  4569  4570  4586  4587  4588  4589  4590  4592  4593  4612  4613  4617
        4621  4625  4629  4633  4634  4640  4641  4642  4643  4662  4663  4669  4670  4672  4675
        4680  4690  4702  4703  4706  4707  4710  4711  4715  4721  4728  4735  4737  4738  4739
        4783  4790  4791  4795  4796  4821  4827  4848  4851  4865  4870  4874  4877  4884  4890
        4896  4897  4900  4904  4907  4911  4914  4919  4927  4937  4938  4954  4955  4968  4970
        4975  4985  5025  5026  5027  5028  5050  5051  5052  5053  5054  5058  5060  5061  5080
        5081  5082  5083  5084  5100  5104  5131  5132  5137  5145  5159  5199  5200  5207  5211
        5216  5217  5219  5221  5227  5229  5238  5246  5252  5253  5257  5258  5262  5263  5264
        5265  5268  5269  5275  5299  5300  5301  5302  5303  5304  5305  5309  5310  5314  5334
        5340  5341  5342  5343  5344  5347  5348  5381  5389  5390  5391  5397  5404  5423  5424
        5425  5426  5427  5439  5451  5452  5453  5454  5455  5461  5469  5470  5479  5485  5493
        5494  5495  5496  5497
MOVB    1427  2036  2047  2115  2124  2152  2162  3167  3174  3175  3181  3252  3255  3278  3885
        3996  4477  4489  4705  4709  4780  4829  4838  4924  4958  4977  4982  4989  4995  5016
        5064  5142  5169  5177  5194  5195  5197  5308  5312  5326  5329  5338  5382  5383  5386
        5387  5388  5392  5395  5396  5415  5487
NEG     5307  5393
NEGB    3188
NOP     3759  3784  3785  3786
RESET   1726  3782
ROL     1537  1539  1553  1794  1872  1914  3840  3856  3938  3957  4065  4067  4511  4615  4638
        5071  5073  5075  5399  5401  5402  5403  5405  5462  5463  5464  5465  5466  5472  5478
        5482
ROLB    4104  4105  4106  4107  4108  4109  4110  4111  4112  4113  4114  4115  4116  4117  4118
        4119  4120  4121
ROR      844   853  4028  4029  4081  4207  4208  4218  4219  4616
RTI      914  4389  4960  5029  5085  5147  5349  5428  5446
RTS     3713  3820  3920  4005  4010  4048  4070  4089  4100  4122  4154  4191  4283  4330  4351
        4413  4441  4529  4543  4571  4644  4736  4800  4853  4941  5107  5186  5254  5266  5276
        5500
SBC     4601
SEC     2291  3678
SOB     5473
```

```
SUB     2009  2094  2228  2229  3305  3320  3661  3698  3701  3979  4027  4037  4179  4206  4217
        4240  4245  4308  4310  4313  4319  4320  4321  4324  4325  4326  4568  4600  4828  4878
        5214  5259  5315
SWAB    3222
TST      818   821   865  1439  1469  1527  1548  1598  1608  1622  1630  1654  1667  1714  1724
        1732  1738  1756  1775  1777  1790  1831  1888  1890  1925  1932  1946  1967  3146  3178
        3184  3225  3657  3717  3718  3722  3740  3830  3902  3933  4007  4025  4056  4201  4275
        4291  4314  4340  4348  4365  4367  4373  4375  4395  4399  4410  4418  4424  4438  4466
        4479  4481  4492  4498  4506  4517  4537  4577  4579  4671  4697  4716  4722  4742  4748
        4771  4773  4843  4849  4931  4980  4993  5024  5079  5086  5144  5152  5173  5209  5230
        5234  5242  5244  5260  5270  5320  5330  5410  5435  5457
TSTB    1593  1842  3279  4491  4598  4605  4684  4902  4956  5127  5175  5201  5212  5225  5240
        5322  5336
.ASCII   961   962  5558  5581
.ASCIZ   960   963  1459  3789  4942  5032  5509  5516  5519  5523  5527  5529  5530  5539  5544
        5551  5565  5572  5575  5577  5587  5591  5595  5600  5604  5610  5615  5622  5628  5634
        5638  5641  5647  5652  5657  5663  5669  5675  5681  5687  5694  5702  5712  5723  5731
        5742  5747  5754  5759  5766  5774  5777  5780  5785  5793  5796  5800  5803  5806  5811
        5816  5822  5828  5831  5836  5839
.BLKB   5033
.BLKW   1273  5354
.BYTE    928   929   934   935   950   951   952   953   982   983   993   994  1001  1002  1004
        1005  1007  1008  1010  1011  1012  1013  1121  1122  1123  3683  3684  3792  4623  4624
        4631  4632  4785  4786  4840  4841  4929  4930  5030  5031  5277  5278  5279  5429  5430
        5431  5432  5501  5502  5846  5848  5850  5852  5854  5856  5859
.ENABL     1   592
.END    5865
.ENDC    593   598   603   613   615   616   617   625   711   725   740   751   762   774   787
         798   804   806   809   814   880   892   902   912   914   918   919   926   928   954
         958   959   960   961   965   966   993  1001  1004  1007  1010  1011  1012  1013  1016
        1017  1018  1019  1020  1021  1022  1023  1024  1025  1026  1027  1028  1029  1030  1031
        1032  1033  1034  1037  1038  1042  1049  1060  1092  1094  1141  1145  1160  1162  1195
        1197  1214  1224  1276  1421  1424  1433  1434  1436  1451  1455  1459  1498  1507  1575
        1578  1644  1647  1677  1681  1691  1692  1704  1705  1711  1712  1718  1721  1736  1737
        1742  1751  1762  1766  1808  1811  1848  1850  1959  1962  1974  1975  1981  1982  1988
        1989  1998  1999  2003  2004  2015  2016  2019  2020  2021  2022  2028  2029  2032  2033
        2040  2041  2052  2053  2056  2057  2059  2060  2066  2067  2070  2071  2079  2080  2092
        2093  2097  2098  2100  2101  2107  2108  2111  2112  2118  2119  2129  2130  2133  2134
        2136  2137  2143  2144  2147  2148  2155  2156  2167  2168  2171  2172  2174  2177  2178
        2179  2180  2186  2187  2190  2191  2196  2197  2199  2200  2201  2202  2205  2206  2214
        2215  2218  2219  2231  2233  2234  2235  2236  2239  2240  2244  2251  2252  2255  2256
        2258  2259  2260  2263  2264  2275  2276  2279  2280  2282  2283  2284  2287  2288  2299
        2300  2303  2304  2305  2306  2307  2313  2314  2344  2345  2347  2349  2358  2359  2366
        2367  2374  2375  2382  2383  2390  2391  2393  2395  2405  2406  2408  2410  2419  2420
        2427  2428  2435  2436  2443  2444  2451  2452  2453  2454  2455  2461  2462  2468  2469
        2471  2473  2482  2483  2489  2490  2496  2497  2503  2504  2511  2512  2514  2516  2526
        2527  2535  2536  2544  2545  2554  2555  2556  2557  2558  2564  2565  2571  2572  2575
        2577  2586  2587  2593  2594  2600  2601  2607  2608  2615  2616  2618  2620  2630  2631
        2639  2640  2648  2649  2658  2659  2660  2661  2662  2668  2669  2675  2676  2678  2680
        2690  2691  2697  2698  2704  2705  2711  2712  2723  2724  2726  2728  2738  2739  2747
        2748  2756  2757  2763  2764  2772  2773  2781  2782  2788  2789  2797  2798  2806  2807
        2813  2814  2822  2823  2831  2832  2843  2944  2846  2847  2848  2854  2855  2861  2862
        2864  2866  2876  2877  2883  2884  2890  2891  2897  2898  2909  2910  2912  2914  2924
        2925  2933  2934  2942  2943  2949  2950  2958  2959  2967  2968  2974  2975  2983  2984
        2992  2993  2999  3000  3008  3009  3017  3018  3029  3030  3031  3032  3033  3039  3040
        3053  3054  3056  3058  3070  3071  3077  3078  3085  3086  3088  3090  3100  3101  3104
```

```
        3106  3118  3119  3125  3126  3133  3134  3135  3136  3141  3142  3145  3146  3171  3172
        3193  3194  3209  3210  3217  3218  3237  3238  3250  3251  3267  3268  3276  3277  3291
        3292  3293  3296  3297  3316  3317  3324  3325  3327  3328  3350  3351  3357  3358  3370
        3371  3375  3376  3377  3378  3400  3401  3407  3408  3420  3421  3425  3426  3427  3428
        3450  3451  3457  3458  3471  3472  3476  3477  3478  3479  3501  3502  3508  3509  3521
        3522  3526  3527  3528  3529  3551  3552  3558  3559  3571  3572  3576  3577  3578  3579
        3601  3602  3608  3609  3622  3623  3627  3628  3629  3630  3638  3639  3645  3646  3712
        3713  3730  3750  3753  3754  3755  3757  3760  3765  3768  3769  3778  3779  3788  3789
        3792  3793  3796  3802  3824  3826  3923  3928  4013  4018  4051  4054  4073  4075  4092
        4094  4099  4100  4102  4104  4125  4127  4157  4159  4177  4178  4193  4195  4286  4288
        4333  4337  4354  4358  4371  4372  4383  4384  4392  4395  4416  4418  4430  4431  4436
        4437  4444  4448  4488  4489  4532  4534  4547  4550  4574  4577  4624  4625  4632  4633
        4647  4653  4658  4664  4666  4677  4680  4683  4684  4686  4688  4695  4699  4704  4706
        4710  4801  4802  4803  4809  4819  4827  4832  4833  4835  4843  4846  4853  4854  4855
        4877  4944  4945  4948  4961  4962  4970  4972  5012  5033  5034  5036  5045  5094  5096
        5100  5110  5142  5192  5195  5198  5240  5277  5286  5356  5441  5442  5506  5509  5645
        5647  5790  5792  5844  5846
.EQUIV   625   626   628   643   644   673   674   675   676   677   678   679   680   681   682
         701   702   703   704   705   706   707   708   709   710
.EVEN    971  1014  1124  1459  4943  5034  5280  5861
.IF      593   594   603   612   614   615   616   617   623   683   711   740   751   762   774
         784   797   802   804   808   813   879   892   902   910   912   914   918   925   927
         954   958   959   960   964   965   993  1001  1004  1007  1010  1011  1012  1013  1016
        1017  1018  1019  1020  1021  1022  1023  1024  1025  1026  1027  1028  1029  1030  1031
        1032  1033  1034  1035  1037  1041  1048  1059  1091  1093  1140  1144  1159  1161  1194
        1196  1213  1223  1420  1423  1429  1433  1434  1436  1446  1455  1458  1497  1506  1574
        1577  1643  1646  1676  1680  1691  1704  1711  1717  1720  1736  1741  1750  1761  1765
        1807  1810  1847  1849  1958  1961  1973  1975  1981  1984  1988  1998  2002  2003  2015
        2018  2019  2020  2022  2028  2031  2032  2039  2040  2052  2055  2056  2058  2060  2066
        2069  2070  2078  2079  2092  2096  2097  2099  2101  2107  2110  2111  2117  2118  2129
        2132  2133  2135  2137  2143  2146  2147  2154  2155  2167  2170  2171  2173  2176  2177
        2179  2180  2186  2189  2190  2195  2196  2198  2200  2201  2204  2205  2214  2217  2218
        2219  2232  2234  2235  2238  2239  2243  2251  2254  2255  2257  2259  2262  2263  2275
        2278  2279  2281  2283  2286  2287  2299  2302  2303  2304  2306  2309  2313  2343  2344
        2346  2348  2358  2366  2374  2382  2389  2390  2392  2394  2404  2405  2407  2409  2419
        2427  2435  2443  2450  2451  2452  2454  2457  2461  2467  2468  2470  2472  2482  2489
        2496  2503  2510  2511  2513  2515  2526  2535  2544  2553  2554  2555  2557  2560  2564
        2570  2571  2574  2576  2586  2593  2600  2607  2614  2615  2617  2619  2630  2639  2648
        2657  2658  2659  2661  2664  2668  2674  2675  2677  2679  2690  2697  2704  2711  2722
        2723  2725  2727  2738  2747  2756  2763  2772  2781  2788  2797  2806  2813  2822  2931
        2842  2843  2845  2847  2850  2854  2860  2861  2863  2865  2876  2883  2890  2897  2908
        2909  2911  2913  2924  2933  2942  2949  2958  2967  2974  2983  2932  2999  3008  3017
        3028  3029  3030  3032  3035  3039  3052  3053  3055  3057  3070  3077  3084  3085  3087
        3089  3099  3100  3103  3105  3118  3125  3132  3133  3134  3136  3141  3144  3145  3171
        3193  3209  3217  3237  3250  3267  3276  3290  3292  3295  3296  3316  3323  3324  3326
        3328  3350  3353  3357  3370  3374  3375  3376  3378  3400  3403  3407  3420  3424  3425
        3426  3428  3450  3453  3457  3471  3475  3476  3477  3479  3501  3504  3508  3521  3525
        3526  3527  3529  3551  3554  3558  3571  3575  3576  3577  3579  3601  3604  3608  3622
        3626  3627  3628  3630  3638  3641  3645  3712  3729  3749  3753  3754  3755  3756  3757
        3759  3764  3767  3769  3778  3779  3788  3789  3795  3801  3823  3825  3922  3927  4012
        4017  4050  4053  4072  4074  4091  4093  4098  4099  4101  4103  4124  4126  4156  4158
        4177  4192  4194  4285  4287  4332  4336  4353  4357  4370  4371  4383  4391  4394  4415
        4417  4430  4436  4443  4447  4488  4531  4533  4546  4549  4573  4576  4623  4624  4631
        4632  4646  4652  4657  4662  4664  4676  4678  4679  4680  4684  4685  4686  4695  4697
        4705  4707  4712  4801  4802  4808  4818  4822  4830  4832  4833  4836  4843  4846  4853
        4854  4876  4896  4944  4948  4961  4969  4971  4979  5032  5033  5034  5035  5041  5061
```

```
        5095    5099    5109    5133    5191    5195    5198    5225    5256    5285    5355    5441    5505    5508    5644
        5646    5789    5791    5843    5845
.IFF     612     615     616     617     623     798     804     809     814     880     910     919     925     928     954
         965     966    1038    1042    1049    1060    1092    1094    1141    1145    1160    1162    1195    1197    1214
        1224    1433    1498    1507    1575    1578    1644    1647    1677    1681    1691    1704    1711    1718    1721
        1736    1742    1751    1762    1766    1808    1811    1848    1850    1973    1974    1975    1982    1988    1998
        2003    2004    2015    2018    2019    2020    2021    2022    2029    2031    2032    2040    2041    2052    2055
        2056    2058    2059    2060    2067    2069    2070    2079    2080    2092    2096    2097    2099    2100    2101
        2108    2110    2111    2118    2119    2129    2132    2133    2135    2136    2137    2144    2146    2147    2155
        2156    2167    2170    2171    2174    2177    2178    2179    2180    2187    2189    2190    2195    2197    2198
        2199    2200    2202    2204    2205    2214    2217    2219    2232    2233    2234    2236    2238    2239    2244
        2251    2254    2255    2257    2258    2259    2260    2262    2263    2275    2278    2280    2281    2282    2283
        2284    2286    2287    2299    2302    2304    2305    2306    2307    2313    2343    2345    2347    2349    2358
        2366    2374    2382    2389    2390    2393    2395    2404    2405    2408    2410    2419    2427    2435    2443
        2450    2451    2452    2453    2454    2455    2461    2467    2469    2471    2473    2482    2489    2496    2503
        2510    2511    2514    2516    2526    2535    2544    2553    2554    2555    2556    2557    2558    2564    2570
        2572    2575    2577    2586    2593    2600    2607    2614    2615    2618    2620    2630    2639    2648    2657
        2658    2659    2660    2661    2662    2668    2674    2676    2678    2680    2690    2697    2704    2711    2722
        2723    2726    2728    2738    2747    2756    2763    2772    2781    2788    2797    2806    2813    2822    2831
        2842    2843    2845    2846    2847    2848    2854    2860    2862    2864    2866    2876    2883    2890    2897
        2908    2909    2912    2914    2924    2933    2942    2949    2958    2967    2974    2983    2992    2999    3008
        3017    3028    3029    3030    3031    3032    3033    3039    3052    3054    3056    3058    3070    3077    3084
        3085    3088    3090    3099    3100    3104    3106    3118    3125    3132    3133    3134    3135    3136    3142
        3144    3145    3171    3193    3209    3217    3237    3250    3267    3276    3290    3291    3292    3293    3295
        3296    3316    3323    3325    3326    3327    3328    3351    3357    3370    3375    3376    3377    3378    3401
        3407    3420    3425    3426    3427    3428    3451    3457    3471    3476    3477    3478    3479    3502    3508
        3521    3526    3527    3528    3529    3552    3558    3571    3576    3577    3578    3579    3602    3608    3622
        3627    3628    3629    3630    3639    3645    3712    3729    3750    3756    3759    3765    3768    3789    3796
        3802    3824    3826    3923    3928    4013    4018    4051    4054    4073    4075    4092    4094    4099    4100
        4102    4104    4125    4127    4157    4159    4177    4193    4195    4286    4288    4333    4337    4354    4358
        4371    4383    4392    4395    4416    4418    4430    4436    4444    4448    4488    4532    4534    4547    4550
        4574    4577    4624    4632    4633    4647    4677    4680    4686    4712    4801    4803    4808    4822    4853
        4855    4876    4896    4945    4954    4961    4962    4972    5012    5034    5036    5096    5100    5110    5192
        5286    5356    5441    5506    5509    5645    5647    5790    5792    5844    5846
.IFT    1459    4694    4833    4954    5066    5086    5094
.IFTF   1459    4692    4832    4948    5062    5070    5093
.IIF     593     598     603     609     610     611     613     616     617     618     619     620     781     911     964
         968    1434    1455    1456    1494    1495    1517    1518    1562    1563    1570    1571    1588    1589    1601
        1602    1611    1612    1633    1634    1670    1671    1814    1815    1820    1821    1823    1826    1828    1829
        1855    1856    1860    1882    1883    1887    1927    1928    1939    1940    1944    3748    3749    3754    3760
        3770    3772    3775    3777    3789    3793    4184    4185    4360    4361    4378    4379    4457    4458    4495
        4496    4520    4521    4582    4583    4603    4610    4618    4622    4626    4630    4653    4654    4655    4656
        4657    4658    4693    4694    4710    4733    4734    4777    4778    4784    4801    4802    4809    4810    4811
        4812    4813    4825    4834    4846    4854    4864    4871    4875    4887    4889    4893    4895    4905    4908
        4912    4915    4928    4934    4940    4948    4976    4984    4991    4997    5002    5008    5010    5014    5018
        5023    5024    5034    5059    5089    5091    5094    5105    5154    5191    5346    5417    5489    5490    5499
        5500
.IRP     774     886     902    1037    1973    2020    2058    2099    2135    2177    2198    2232    2257    2281    2304
        2452    2555    2659    2845    3030    3134    3219    3240    3258    3271    3290    3326    3376    3426    3477
        3527    3577    3628    3759    4058    4069    4076    4087    4160    4188    4199    4215    4280    4311    4318
        4362    4387    4407    4412    4422    4440    4449    4468    4476    4503    4513    4522    4535    4542    4586
        4612    4633    4640    4662    4712    4818    4853    5052    5082    5199    5200    5221    5252    5253    5257
        5265    5268    5275    5299    5340    5451    5493
.LIST      1     592     604     616     725     774     781     918     954     955     956     957     958     965     968
        1419    1425    1436    1455    1459    1957    1963    1973    1988    2020    2032    2058    2070    2099    2111
        2135    2147    2177    2190    2198    2205    2232    2239    2257    2263    2281    2287    2304    2313    2452
```

```
             2461    2555    2564    2659    2668    2845    2854    3030    3039    3134    3145    3290    3296    3326    3357
             3376    3407    3426    3457    3477    3508    3527    3558    3577    3608    3628    3645    3760    4657    4846
             4961
.MACRO          1     592     617     918    1973    2198    3134    3326    3376    3426    3477    3527    3577    3628    4646
.MCALL        592     725     965    1436
.MEXIT       1036
.NLIST          1     592     604     616     725     774     781     918     954     955     956     957     958     965     968
             1419    1425    1436    1455    1459    1957    1963    1973    1988    2020    2032    2058    2070    2099    2111
             2135    2147    2177    2190    2198    2205    2232    2239    2257    2263    2281    2287    2304    2313    2452
             2461    2555    2564    2659    2668    2845    2854    3030    3039    3134    3145    3290    3296    3326    3357
             3376    3407    3426    3457    3477    3508    3527    3558    3577    3608    3628    3645    3760    4657    4846
             4961
.PAGE         807     879     918     964    1740    1846    2020    2257    2304    2452    2555    2659    3030    3134    3290
             3376    3426    3477    3527    3577    3628    3726    3749    4049    4101    4155    4352
.REM             1
.REPT          781     954    1063    1420    1423    1958    1961
.SBTTL         605     621     728     775     785     799     881     920     967    1039    1061    1193    1212    1277    1418
             1642    1675    1740    1846    1972    1973    2020    2058    2099    2135    2172    2177    2198    2232    2257
             2281    2304    2452    2555    2659    2845    3030    3134    3290    3325    3326    3376    3426    3477    3527
             3577    3628    3726    3751    3793    3794    4049    4155    4352    4545    4648    4804    4856    4946    5037
             5111    5193    5287    5357    5448    5504    5643
.TITLE        593
.WORD         781     782     783     794     795     805     876     877     878     911     913     927     930     931     932
              933     936     937     938     939     940     941     942     943     944     945     954     955     956     957
              973     974     975     976     977     978     979     980     984     985     986     999    1003    1006    1009
             1015    1016    1017    1018    1019    1020    1021    1022    1023    1024    1025    1026    1027    1028    1029
             1030    1031    1032    1033    1034    1053    1054    1055    1056    1057    1058    1063    1064    1065    1066
             1067    1068    1069    1070    1071    1072    1073    1074    1075    1076    1077    1078    1079    1080    1081
             1082    1083    1084    1085    1086    1087    1094    1095    1097    1098    1100    1101    1103    1104    1107
             1108    1110    1111    1113    1114    1116    1117    1118    1125    1127    1128    1130    1131    1133    1134
             1136    1137    1138    1191    1456    1494    1517    1562    1570    1588    1601    1611    1633    1670    1691
             1704    1711    1736    1814    1820    1828    1855    1882    1927    1939    1984    1998    2015    2031    2052
             2069    2092    2110    2129    2146    2167    2189    2204    2214    2238    2251    2262    2275    2286    2299
             2309    2358    2366    2374    2382    2419    2427    2435    2443    2457    2482    2489    2496    2503    2526
             2535    2544    2560    2586    2593    2600    2607    2630    2639    2648    2664    2690    2697    2704    2711
             2738    2747    2756    2763    2772    2781    2788    2797    2806    2813    2822    2831    2850    2876    2883
             2890    2897    2924    2933    2942    2949    2958    2967    2974    2983    2992    2999    3008    3017    3035
             3070    3077    3118    3125    3144    3171    3193    3209    3217    3237    3250    3267    3276    3295    3316
             3353    3370    3403    3420    3453    3471    3504    3521    3554    3571    3604    3622    3641    3712    3729
             3748    3764    3767    3770    3777    4177    4184    4230    4231    4259    4260    4296    4297    4306    4307
             4360    4371    4378    4383    4430    4436    4457    4488    4495    4520    4582    4603    4610    4733    4777
             4825    4834    4864    4887    4889    4893    4895    4934    4940    4984    4991    4997    5002    5008    5010
             5014    5018    5023    5089    5091    5093    5106    5139    5185    5223    5346    5417    5433    5436    5441
             5489    5499
```

```
ERRORS DETECTED:  0
DEFAULT GLOBALS GENERATED:  0

*DZQMCB,DZQMCB.SEQ/SOL/CRF/DS:ERFZ/EN:ABS=DSKM:SYSMAC.SML,DSKM:DZQMCB.P11
RUN-TIME:  74 82 12 SECONDS
RUN-TIME RATIO:  311/171=1.8
CORE USED:  36K  (71 PAGES)
```

Spooler runtime 24 Seconds, 101 KCS, 717 disk reads, 4 disk writes, 150 pages