KD11-K

11/6X MEMORY MGT DIAG

MD-11-DQKTA-A

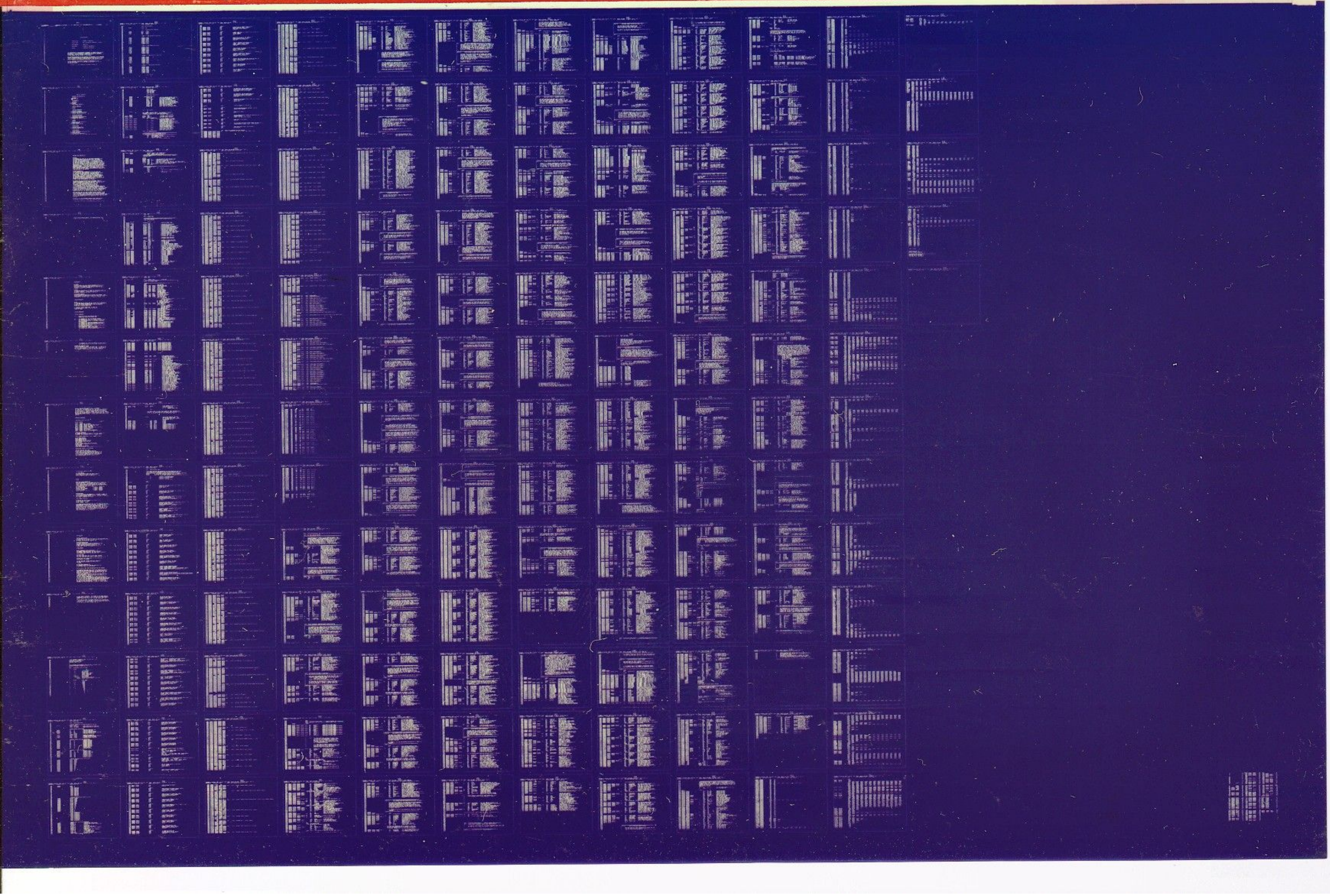
EP-DQKTA-A-DL-A

FICHE 1 OF 1

APR 1977

digital

MADE IN USA



EOF1D[illegible]SB01 MA[illegible]-DQKTA?RO32GP 11/6X MEM. MGMT BIBG. MACY11H8R1D0B6AAS80-FEB-77 10:37008[illegible] 770323
DQKTAA.P11 07-FEB-77 10:30

.REM @

IDENTIFICATION

PRODUCT CODE:	MAINDEC-11-DQKTA-A-D
PRODUCT NAME:	PDP-11/6X MEMORY MANAGEMENT DIAGNOSTIC
DATE CREATED:	MARCH,1977
MAINTAINER:	DIAGNOSTIC ENGINEERING
AUTHOR:	DIAGNOSTIC ENGINEERING

TABLE OF CONTENTS

1. ABSTRACT AND DESCRIPTION

1.1 ABSTRACT

THIS PROGRAM WILL TEST ALL OF THE MEMORY MANAGEMENT LOGIC, INCLUDING THE STACK LIMIT REGISTER LOGIC, AND ENABLE THE FIELD SERVICE REPRESENTATIVE TO ISOLATE THE DETECTED FAILURES TO A REPLACABLE MODULE. IT IS ASSUMED THAT THE CPU HAS BEEN TESTED, OR IS KNOWN TO BE FUNCTIONING CORRECTLY, AND THAT THE PROGRAM IS STARTED FROM ADDRESS 200. THE 11/6X SERIES CACHE IS TURNED OFF FOR THE FIRST PASS OF THE PROGRAM (SEE SECTION 4.3) AND IS TURNED BACK ON FOR THE SECOND AND SUBSEQUENT PASSES. THIS WILL PROVIDE THE EARLIEST DETECTION OF MEMORY MANAGEMENT RELATED ERRORS AND ENABLE LOOPING ON THE ERROR INVOLVING MINIMUM LOGIC. THIS PROGRAM MAY ALSO EXPOSE FAULTS THAT ARE ON THE INTERFACE BETWEEN MEMORY MANAGEMENT AND OTHER SECTIONS OF THE COMPUTER.

1.2 PROGRAM DESCRIPTION

THIS PROGRAM HAS BEEN SEGMENTED IN THE FOLLOWING WAY:
ALL DATA TABLES, ERROR MESSAGES, AND TRAP HANDLERS RESIDE IN LOW CORE (VIRTUAL PAGES 0 IE. ADDRESSES 001100 THRU 017776). RIGHT NOW THE END OF THE TRAP HANDLERS IS AROUND 015500, SO THERE IS SOME ROOM FOR FUTURE EXPANSION. THE TEST CODE STARTS AT VIRTUAL PAGE 1 (ADDRESS 020000) AND EXPANDS TOWARD PAGE 3 (ADDRESS 060000). THE END OF THE PROGRAM IS NOW AROUND ADDRESS 044300, SO MODIFICATIONS CAN BE MADE WITHOUT RE-SEGMENTING THE PROGRAM.

THE REASON FOR THIS SEGMENTATION IS TWO-FOLD, FIRST IT ENABLES THE OPERATOR TO TELL FROM THE ADDRESS DISPLAYED EXACTLY WHERE THE PROGRAM HAS HALTED OR "HUNG-UP". THAT IS, DID IT HALT IN THE TRAP ROUTINE BECAUSE OF A CONDITION IMPOSSIBLE TO RECOVER FROM (ON PAGE 0), OR DID IT GET "HUNG-UP" IN THE TEST CODE ON PAGE 1 OR 2.
THE OTHER REASON IS THAT CERTAIN MEMORY MANAGEMENT FUNCTIONS LOCK UP THE VIRTUAL PC OF THE INSTRUCTION AND THE PROGRAM, IN ORDER TO OPERATE PROPERLY, ONE MUST KNOW WHERE IT IS AT ALL TIMES. IT SEEMS MUCH SIMPLER FOR THE CODE TO START AT A PREDETERMINED BOUNDARY SO THAT IF THE MESSAGES CHANGE OR A NEW SUBROUTINE IS ADDED THE PAGE THAT THE CODE IS ON WILL REMAIN THE SAME.

EACH TEST WILL SET THE LOOP ON ERROR POINTER ($LPERR) TO THE MINIMUM NECESSARY SETUP CODE, IF ANY, FOR THE FUNCTION UNDER TEST. IF ON THE 11/6X AN "EXTERNAL SYNC PULSE" ON THE BACK PLANE IS DESIRED, THE MICROBREAK REGISTER SHOULD BE LOADED FROM THE CONSOLE WITH A MICRO-ADDRESS USED IN THE INSTRUCTION THAT TESTS EACH NEW FUNCTION.

SECTION 8.4 OF THIS DOCUMENT CONTAINS SOME IDEAS ON HOW TO EFFECTIVELY UTILIZE IT TO MAKE FAULT ISOLATION EASIER.

IT SHOULD BE NOTED THAT THIS PROGRAM DOES NOT CHECK OUT THE

```
150
151                                     CONSOLE OR THE CONSOLE CABLES.  THE PROGRAM ASSUMES THAT
152                                     THOSE COMPONENTS HAVE BEEN TESTED OR ARE KNOWN TO BE GOOD.
```

2. REQUIREMENTS

2.1 EQUIPMENT

THE BASIC PDP-11/40 (WITH MEMORY MANAGEMENT-KT11-D) OR A PDP-11/6X SERIES COMPUTER, INCLUDING AN OPERATING CPU AND MEMORY. A CONSOLE TERMINAL IS ALSO NEEDED FOR ERROR MESSAGES, AND END OF PASS REPORTS.

2.2 STORAGE

THIS PROGRAM REQUIRES 12K OF MEMORY TO LOAD AND AT LEAST 16K OF MEMORY TO RUN IN. IT WILL SCAN MEMORY FROM 12K TO 124K ON 2K BOUNDARIES, REPORTING ANY HOLES IT MAY FIND.

2.3 PRELIMINARY PROGRAMS

THE CPU DIAGNOSTICS SHOULD BE RUN BEFORE THIS PROGRAM. MAIN MEMORY SHOULD BE SCANNED FOR AT LEAST THE FIRST 16K TO SEE THAT A PROGRAM WILL EXECUTE CORRECTLY BEFORE ANY PROGRAM IS RUN.

3. LOADING PROCEDURE

3.1 METHOD

THIS PROGRAM CAN BE LOADED FROM ANY DEVICE THAT IS SUPPORTED BY XXDP, AND SHOULD BE LOADED USING THE XXDP PROCEDURE FOR THAT DEVICE OR IT CAN BE LOADED DIRECTLY USING THE ABSOLUTE LOADER AND THE BINARY PAPER TAPE.

IF LOADING A BINARY PAPER TAPE, BE SURE THE SWITCH REGISTER IS CLEARED AFTER THE ABSOLUTE LOADER IS LOADED.

4. STARTING PROCEDURE

4.1 STARTING ADDRESSES

NOTE: CHECK THAT SWITCH REGISTER WAS ZERO IF PROGRAM WAS LOADED FROM PAPER TAPE.

200	THIS ADDRESS WILL RUN THE COMPLETE PROGRAM
204	THIS ADDRESS WILL START THE PROGRAM AT ENTRY POINT 2 TEST THE READ/WRITE BITS IN THE MEMORY MANAGEMENT STATUS REGISTERS
210	THIS ADDRESS WILL START THE PROGRAM AT ENTRY POINT 3 PAGE ADDRESS AND PAGE DESCRIPTOR REGISTER TESTS
214	THIS ADDRESS WILL START THE PROGRAM AT ENTRY POINT 4 RELOCATION AND ADDER TESTS
220	THIS ADDRESS WILL START THE PROGRAM AT ENTRY POINT 5 MEMORY MANAGEMENT ABORTS LOGIC TESTS
224	THIS ADDRESS WILL START THE PROGRAM AT ENTRY POINT 6 W BIT LOGIC TEST AND DUAL MAPPING TESTS

```
                                        230     THIS ADDRESS WILL START THE PROGRAM AT ENTRY POINT 7
                                                MOVE FROM AND MOVE TO PERVIOUS MODE INSTRUCTION TESTS

                                4.2     PROGRAM AND OPERATOR ACTION
                                        AFTER THE PROGRAM IS LOADED, THE FIRST TIME IT IS RUN IT WILL
                                        IDENTIFY ITSELF AND RUN A QUICK VERIFY PASS.  AT THE END OF
                                        EACH PASS THE PROGRAM WILL TYPE OUT THE PASS NUMBER AND THE
                                        TOTAL NUMBER OF ERRORS FOUND ON THAT PASS, AS FOLLOWS:

                                        "END PASS #     1   TOTAL ERRORS SINCE LAST REPORT     0"
```

4.3 SPECIAL STARTING PROCEDURE ON THE 11/6X SERIES

FOR THE FIRST TIME THROUGH THE TESTS THE CACHE WILL BE TURNED OFF (BITS 2 & 3 - FORCE MISS 0 & 1 - ARE SET IN THE CACHE CONTROL REGISTER.). CACHE IS TURNED ON AFTER THE FIRST PASS AND LEFT ON FOR THE REMAINING PASSES (BITS 2 & 3 OF THE CONTROL REGISTER ARE CLEARED AND BIT 7 - PARITY ERROR ABORT BIT - IS SET ENABLING CACHE PARITY ERRORS TO BE REPORTED IN THE MEMORY ERROR REGISTER.)

5. OPERATING PROCEDURE

5.1 OPERATIONAL SWITCH SETTINGS

SWITCH	DISPLAY	
SW15=1	100000	HALT ON ERROR
SW14=1	040000	LOOP ON THE TEST THAT YOU ARE IN
SW13=1	020000	INHIBIT ALL ERROR TYPE OUTS
SW12=1	010000	INHIBIT TRACE TRAP
SW11=1	004000	INHIBIT ITERATIONS AFTER FIRST PASS
SW10=1	002000	RING BELL ON ERROR
SW09=1	001000	LOOP ON ERROR
SW08=1	000400	LOOP ON TEST IN SWR<06:00>
SW07=1	000200	INHIBIT MULTIPLE ERROR TYPE OUTS

5.2 SUBROUTINE ABSTRACTS

ALL SUBROUTINE ABSTRACTS APPEAR IN THE CODE, BEFORE THEIR EXPANSION, THE FOLLOWING IS A LIST OF THEIR TITLES.

5.2.1 MACRO LIBRARY SUBROUTINES (FOUND IN MOST PROGRAMS)

END OF PASS ROUTINE
SCOPE HANDLER ROUTINE
ERROR HANDLER ROUTINE
ERROR MESSAGE TYPE OUT ROUTINE
SAVE & RESTORE R0-R5 ROUTINES
TYPE ROUTINE
BINARY TO OCTAL (ASCII) AND TYPE ROUTINE
CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
TRAP DECODER
POWER DOWN AND UP ROUTINE
DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE

5.2.2 SUBROUTINES UNIQUE TO THIS PROGRAM

TURN OFF AND SAVE THE T-BIT
RESTORE T-BIT TO ITS PREVIOUS CONDITION
CLEAR 8 PAR'S OR PDR'S STARTING FROM ADDRESS IN R5
CLEANUP LOCATIONS THAT HOLD LOGICAL 'AND' AND 'OR'
P.A.R. OR P.D.R. ADDRESS TIMED OUT WHEN REFERENCED
DUAL ADDRESSING WHEN LOADING A P.A.R. OR P.D.R.
COUNT PATTERN ERROR IN P.A.R.'S OR P.D.R.'S

```
                                5.2.3   TRAP AND ABORT HANDLER ROUTINES

                                        CPU TRAP HANDLER
                                        CACHE/MEMORY-PARITY TRAPS AND ABORTS HANDLER
                                        MEMORY MANAGEMENT TRAPS AND ABORTS HANDLER
                                        TRAP ROUTINES FOR ABORT IN USER MODE



                                6.      ERRORS
                                        ------

                                6.1     ERROR HALTS AND DESCRIPTION
                                        WHEN THE PROGRAM DETECTS AN ERROR CONDITION IT ISSUES AN
                                        "ERROR" (EMT) CALL.  THIS CAUSES THE CPU TO TRAP TO THE
                                        "ERROR HANDLER ROUTINE" WHICH PRINTS OUT THE ERROR MESSAGE,
                                        IF ANY, AND CHECKS THE SWITCH REGISTER FOR THE MODE SELECTED.
                                        THE PROGRAM WILL REACT AS FOLLOWS:
                                        HALT ON THE ERROR                   IF SW15=1, (100000)
                                        INHIBIT ERROR TYPE OUT              IF SW13=1, (020000)
                                        RING BELL ON THE ERROR              IF SW10=1, (002000)
                                        LOOP ON THE ERROR                   IF SW09=1, (001000)
                                        INHIBIT MULTIPLE TYPE OUTS          IF SW07=1, (000200)

                                6.2     ERROR RECOVERY
                                        IF SWITCH 09 IS UP, THE PROGRAM WILL LOOP BACK TO WHERE THE
                                        LOOP ON ERROR POINTER ($LPERR) IS SET.  THIS WILL PROVIDE
                                        THE TIGHTEST POSSIBLE SCOPING LOOP, AND PROVIDES ALL NECESSARY
                                        SETUP CODE TO RECREATE THE ERROR.

                                6.3     SAMPLE ERROR TYPE OUTS

                                        UNEXPECTED CPU TRAP OR ABORT THRU "ERRVEC" (004)
                                        CPU ERR TESTNO PC AT ABORT
                                        000020  000033   XXXXXX

                                        THIS ERROR MESSAGE INDICATES THAT THE CPU TIMED OUT OVER THE
                                        UNIBUS WHEN NO TIMEOUTS WERE EXPECTED.  THE CONTENTS OF THE CPU
                                        ERROR REGISTER, IF ON AN 11/6X, ARE SHOWN UNDER "CPU ERR",
                                        SHOWING THAT IN FACT A UNIBUS TIMEOUT DID OCCUR (BIT 4 IS SET).
                                        IF ON AN 11/40, JUST THE TEST NO. AND PC AT THE TIME OF THE
                                        ABORT ARE TYPED OUT.  THIS TEST HAPPENS TO BE CHECKING THE
                                        CARRY PROPAGATION, THAT IS DETERMINED FROM LOOKING AT THE
                                        INDEX AT THE FRONT OF THE LISTING.
```

7. RESTRICTIONS

7.1 STARTING RESTRICTIONS

IF A STARTING POINT OTHER THAN "200" IS USED AND ERRORS ARE REPORTED, THEY MAY BE DUE TO LOGIC THAT IS ASSUMED TO BE WORKING AS A RESULT OF TESTS THAT WERE NOT RUN.

7.2 OPERATING RESTRICTIONS

THE "MED" INSTRUCTION IS USED TO DETERMINE IF THE PROGRAM IS EXECUTING ON AN 11/40 OR AN 11/6X. IF A "MED" TRAPS AS A RESERVED INSTRUCTION, IT IS AUTOMATICALLY ASSUMED THE PROGRAM IS ON AN 11/40.

8. MISCELLANEOUS

8.1 MEMORY MANAGEMENT STATUS REGISTERS

THE SYMBOLS "MMR0,MMR1,MMR2" ARE USE THROUGHOUT THE PROGRAM. THESE REPRESENT THE THREE MEMORY MANAGEMENT STATUS REGISTERS IN THE 11/6X OR THE 11/40.

8.2 EXECUTION TIME

THE TIMES NOTED BELOW ARE "11/6X" TIMES. PROGRAM EXECUTION TIMES WILL BE LONGER IN EXECUTING ON AN 11/40.

THE RUN TIME FOR A SINGLE PASS WITH NO ITERATIONS IS APPROXIMATELY 10 SECONDS.

THE RUN TIME FOR A SINGLE PASS WITH ITERATIONS AND TRACE TRAPPING ENABLED IS APPROXIMATELY 3 MINUTES.

8.3 ACT/APT/XXDP COMPATABILITY

THE PROGRAM IS FULLY ACT AND APT COMPATABLE AND IS SUPPORTED UNDER THE XXDP PACKAGE.

8.4 HINTS ON HOW TO GET MORE INFORMATION FROM THE PROGRAM

IF AN ERROR OCCURS THE FIRST THING THAT SHOULD BE NOTED IS WHAT PASS DID THE ERROR OCCUR ON. IF THE PASS HAS AN ODD NUMBER AND SWITCH 12 IS DOWN THEN THE ERROR MIGHT BE T-BIT SENSITIVE. TRY TO RUN AGAIN WITH SWITCH 12 UP, THIS WILL INHIBIT T-BIT TRAPPING.
IF THE PASS NUMBER IS GREATER THAN ONE THE ERROR MIGHT BE ITERATION SENSITIVE. TRY TO RUN AGAIN WITH SWITCH 11 UP, THIS WILL INHIBIT ITERATIONS.
NOW THAT YOU HAVE DETERMINED HOW TO MAKE THE MACHINE FAIL LOOK IN THE INDEX AT THE FRONT OF THE LISTING TO FIND THE TITLE OF THE TEST THAT WAS RUNNING WHEN THE ERROR CONDITION OCCURRED.
GO TO THE LISTING AND READ THE PARAGRAPH AT THE BEGINNING OF THE TEST SO THAT YOU KNOW WHAT THE TEST IS TRYING TO DO.
NOW READ THE ERROR MESSAGE AND IF THERE IS A COLUMN LABELED "ERRORPC" GO TO THAT LOCATION IN THE TEST. THIS IS THE PC

```
                                                    OF THE "ERROR" STATEMENT.  THE NUMBER IS THE ERROR MESSAGE
                                                    NUMBER AND IN THE FRONT OF THE LISTING IS THE "ERROR MESSAGE
                                                    POINTER TABLE" WHICH WILL TELL YOU WHAT WORDS WERE TYPED
                                                    OUT.
                                                    IF YOU WANT TO SCOPE THIS ERROR CONDITION, PUT UP SWITCH 09
                                                    (LOOP ON ERROR) OR IF YOU WANT TO LOOP ON THE ENTIRE TEST PUT
                                                    UP SWITCH 14.  YOU WILL PROBABLY WANT TO INHIBIT THE ERROR TYPE
                                                    OUT AT THIS POINT, SWITCH 13 WILL DO THAT.
```

```
                        .TITLE  MAINDEC-11-DQKTA-A  PDP 11/6X MEM. MGMT. DIAG.
                        ;*COPYRIGHT (C) OCTOBER,1976
                        ;*DIGITAL EQUIPMENT CORP.
                        ;*MAYNARD, MASS. 01754
                        ;*
                        ;*
                        ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
                        ;*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
                        ;*
                        .SBTTL  OPERATIONAL SWITCH SETTINGS
                        ;*
                        ;*      SWITCH              USE
                        ;*      ------          --------------------
                        ;*        15            HALT ON ERROR
                        ;*        14            LOOP ON TEST
                        ;*        13            INHIBIT ERROR TYPEOUTS
                        ;*        12            INHIBIT TRACE TRAP
                        ;*        11            INHIBIT ITERATIONS
                        ;*        10            BELL ON ERROR
                        ;*         9            LOOP ON ERROR
                        ;*         8            LOOP ON TEST IN SWR<6:0>
                        ;*         7            INHIBIT MULTIPLE ERROR TYPEOUTS
                        ;*
                        ;*      SWITCH          OCTAL VALUE
                        ;*      ------          ------------
                        ;*        15              100000
                        ;*        14               40000
                        ;*        13               20000
                        ;*        12               10000
                        ;*        11                4000
                        ;*        10                2000
                        ;*         9                1000
                        ;*         8                 400
                        ;*         7                 200
                        ;*         6                 100
                        ;*         5                  40
                        ;*         4                  20
                        ;*         3                  10
                        ;*         2                   4
                        ;*         1                   2
                        ;*         0                   1
```

```
                        .SBTTL  BASIC DEFINITIONS

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

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

                        ;*GENERAL PURPOSE REGISTER DEFINITIONS
000000                  R0=     %0              ;;GENERAL REGISTER
000001                  R1=     %1              ;;GENERAL REGISTER
000002                  R2=     %2              ;;GENERAL REGISTER
000003                  R3=     %3              ;;GENERAL REGISTER
000004                  R4=     %4              ;;GENERAL REGISTER
000005                  R5=     %5              ;;GENERAL REGISTER
000006                  R6=     %6              ;;GENERAL REGISTER
000007                  R7=     %7              ;;GENERAL REGISTER
000006                  SP=     %6              ;;STACK POINTER
000007                  PC=     %7              ;;PROGRAM COUNTER

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

                        ;*"SWITCH REGISTER" SWITCH DEFINITIONS
100000                  SW15=   100000
040000                  SW14=   40000
020000                  SW13=   20000
010000                  SW12=   10000
004000                  SW11=   4000
002000                  SW10=   2000
001000                  SW09=   1000
000400                  SW08=   400
000200                  SW07=   200
000100                  SW06=   100
000040                  SW05=   40
000020                  SW04=   20
000010                  SW03=   10
000004                  SW02=   4
```

```
                    000002          SW01=   2
                    000001          SW00=   1
                                    .EQUIV  SW09,SW9
                                    .EQUIV  SW08,SW8
                                    .EQUIV  SW07,SW7
                                    .EQUIV  SW06,SW6
                                    .EQUIV  SW05,SW5
                                    .EQUIV  SW04,SW4
                                    .EQUIV  SW03,SW3
                                    .EQUIV  SW02,SW2
                                    .EQUIV  SW01,SW1
                                    .EQUIV  SW00,SW0

                                    ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
                    100000          BIT15=  100000
                    040000          BIT14=  40000
                    020000          BIT13=  20000
                    010000          BIT12=  10000
                    004000          BIT11=  4000
                    002000          BIT10=  2000
                    001000          BIT09=  1000
                    000400          BIT08=  400
                    000200          BIT07=  200
                    000100          BIT06=  100
                    000040          BIT05=  40
                    000020          BIT04=  20
                    000010          BIT03=  10
                    000004          BIT02=  4
                    000002          BIT01=  2
                    000001          BIT00=  1
                                    .EQUIV  BIT09,BIT9
                                    .EQUIV  BIT08,BIT8
                                    .EQUIV  BIT07,BIT7
                                    .EQUIV  BIT06,BIT6
                                    .EQUIV  BIT05,BIT5
                                    .EQUIV  BIT04,BIT4
                                    .EQUIV  BIT03,BIT3
                                    .EQUIV  BIT02,BIT2
                                    .EQUIV  BIT01,BIT1
                                    .EQUIV  BIT00,BIT0

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

```

```
                                ;*KT11 VECTOR ADDRESS

000250                          MMVEC=  250

                                ;*KT11 STATUS REGISTER ADDRESSES

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

                                ;*USER "I" PAGE DESCRIPTOR REGISTERS

177600                          UIPDR0= 177600
177602                          UIPDR1= 177602
177604                          UIPDR2= 177604
177606                          UIPDR3= 177606
177610                          UIPDR4= 177610
177612                          UIPDR5= 177612
177614                          UIPDR6= 177614
177616                          UIPDR7= 177616

                                ;*USER "I" PAGE ADDRESS REGISTERS

177640                          UIPAR0= 177640
177642                          UIPAR1= 177642
177644                          UIPAR2= 177644
177646                          UIPAR3= 177646
177650                          UIPAR4= 177650
177652                          UIPAR5= 177652
177654                          UIPAR6= 177654
177656                          UIPAR7= 177656

                                ;*KERNEL "I" PAGE DESCRIPTOR REGISTERS

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

                                ;*KERNEL "I" PAGE ADDRESS REGISTERS

172340                          KIPAR0= 172340
172342                          KIPAR1= 172342
172344                          KIPAR2= 172344
172346                          KIPAR3= 172346
172350                          KIPAR4= 172350
172352                          KIPAR5= 172352
172354                          KIPAR6= 172354
172356                          KIPAR7= 172356

                                        .EQUIV SP,KSP
```

```
                                        .EQUIV SP,USP
                                        .EQUIV STACK,KERSTK
                                        .EQUIV SR0,MMR0
                                        .EQUIV SR1,MMR1
                                        .EQUIV SR2,MMR2
                                        ;
                                        ;ADDITIONAL DEFINITIONS
                                        ;
                000700                  USESTK=STACK-200
                000200                  CRLF=   200             ;CODE FOR CARRIAGE RETURN LINE FEED
                000114                  CACHVEC=114             ;CACHE ERROR INTERRUPT VECTOR
                076600                  MED=    76600           ;OP CODE OF MAINTENANCE EXAM/DEP INST.
                000022                  RWHAM=  022             ;READ CODE FOR WHAMI REG.
                000100                  RLJAM=  100             ;READ CODE FOR LOG JAM REGISTER
                000101                  RLSERV= 101             ;READ CODE FOR LOG SERVICE REGISTER
                000104                  RLFGIN= 104             ;READ CODE FOR LOG FLAG/INTERRUPT REG.
                000105                  RLWHAM= 105             ;READ CODE FOR LOG WHAMI REGISTER
                000222                  WWHAM=  222             ;WRITE CODE FOR WHAMI REG.
                000226                  WCNSSW= 226             ;WRITE CODE FOR CONSOLE SW. REG.
                000305                  WLWHAM= 305             ;WRITE CODE FOR LOG WHAMI REGISTER


                                .SBTTL  TRAP CATCHER

                                        .=0
                000000          ;*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
                                        .=174
000174  000000                  DISPREG: .WORD  0               ;;SOFTWARE DISPLAY REGISTER
000176  000000                  SWREG:   .WORD  0               ;;SOFTWARE SWITCH REGISTER
                                .SBTTL  STARTING ADDRESS(ES)
000200  000137  020000                  JMP     @#STRT1 ;;JUMP TO STARTING ADDRESS OF PROGRAM
                                                                ;RUN ENTIRE PROGRAM (ALL TESTS)
000204  000137  020010                  JMP     STRT2           ;STARTING AT ENTRY POINT 2
                                                                ;MEMORY MANAGEMENT STATUS REGISTERS
000210  000137  020020                  JMP     STRT3           ;STARTING AT ENTRY POINT 3
                                                                ;PAGE ADDRESS AND DESCRIPTOR REGISTERS
000214  000137  020030                  JMP     STRT4           ;STARTING AT ENTRY POINT 4
                                                                ;RELOCATION AND ADDER TESTS
000220  000137  020040                  JMP     STRT5           ;STARTING AT ENTRY POINT 5
                                                                ;MEMORY MANAGEMENT ABORT LOGIC
000224  000137  020050                  JMP     STRT6           ;STARTING AT ENTRY POINT 6
                                                                ;W BIT LOGIC & DUAL MAPPING
000230  000137  020060                  JMP     STRT7           ;STARTING AT ENTRY POINT 7
                                                                ;MFP & MTP LOGIC TESTS
                                .SBTTL  APT PARAMETER BLOCK

                                ;;**************************************************************
                                ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
                                ;;**************************************************************
                000234                  .$X=.   ;;SAVE CURRENT LOCATION
                000024                  .=24    ;;SET POWER FAIL TO POINT TO START OF PROGRAM
000024  000200                  200     ;;FOR APT START UP
                000044                  .=44    ;;POINT TO APT INDIRECT ADDRESS PNTR.
```

```
                                        APT PARAMETER BLOCK

000044  000234                                  $APTHDR ;;POINT TO APT HEADER BLOCK
        000234                                  .=.$X   ;;RESET LOCATION COUNTER
                                        ;;**************************************************************
                                        ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
                                        ;INTERFACE SPEC.

000234                                  $APTHD:
000234  000000                          $HIBTS: .WORD   0       ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
000236  001226                          $MBADR: .WORD   $MAIL   ;;ADDRESS OF APT MAILBOX (BITS 0-15)
000240  000005                          $TSTM:  .WORD   5       ;;RUN TIM OF LONGEST TEST
000242  000015                          $PASTM: .WORD   15      ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
000244  000005                          $UNITM: .WORD   5       ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
000246  000052                                  .WORD   $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
                                        .SBTTL  ACT11 HOOKS

                                        ;;**************************************************************
                                        ;HOOKS REQUIRED BY ACT11
        000250                                  $SVPC=.                 ;SAVE PC
        000046                                  .=46
000046  040762                                  $ENDAD                  ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
        000052                                  .=52
000052  000000                                  .WORD   0               ;;2)SET LOC.52 TO ZERO
        000250                                  .=$SVPC                 ;; RESTORE PC
```

```
                                .SBTTL  COMMON TAGS

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

                        001100          .=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  000                     $AUTOB: .BYTE   0       ;;AUTOMATIC MODE INDICATOR
001135  000                     $INTAG: .BYTE   0       ;;INTERRUPT MODE INDICATOR
001136  000000                          .WORD   0
001140  177570                  SWR:    .WORD   DSWR    ;;ADDRESS OF SWITCH REGISTER
001142  177570                  DISPLAY: .WORD  DDISP   ;;ADDRESS OF DISPLAY REGISTER
001144  177560                  $TKS:   177560          ;;TTY KBD STATUS
001146  177562                  $TKB:   177562          ;;TTY KBD BUFFER
001150  177564                  $TPS:   177564          ;;TTY PRINTER STATUS REG. ADDRESS
001152  177566                  $TPB:   177566          ;;TTY PRINTER BUFFER REG. ADDRESS
001154  000                     $NULL:  .BYTE   0       ;;CONTAINS NULL CHARACTER FOR FILLS
001155  002                     $FILLS: .BYTE   2       ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
001156  012                     $FILLC: .BYTE   12      ;;INSERT FILL CHARS. AFTER A "LINE FEED"
001157  000                     $TPFLG: .BYTE   0       ;;"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
001160  000000                  $REGAD: .WORD   0       ;;CONTAINS THE ADDRESS FROM
                                                        ;;WHICH  ($REG0) WAS OBTAINED
001162  000000                  $REG0:  .WORD   0       ;;CONTAINS (($REGAD)+0)
001164  000000                  $REG1:  .WORD   0       ;;CONTAINS (($REGAD)+2)
001166  000000                  $REG2:  .WORD   0       ;;CONTAINS (($REGAD)+4)
001170  000000                  $REG3:  .WORD   0       ;;CONTAINS (($REGAD)+6)
001172  000000                  $REG4:  .WORD   0       ;;CONTAINS (($REGAD)+10)
001174  000000                  $REG5:  .WORD   0       ;;CONTAINS (($REGAD)+12)
001176  000000                  $TMP0:  .WORD   0       ;;USER DEFINED
001200  000000                  $TMP1:  .WORD   0       ;;USER DEFINED
001202  000000                  $TMP2:  .WORD   0       ;;USER DEFINED
001204  000000                  $TMP3:  .WORD   0       ;;USER DEFINED
001206  000000                  $TMP4:  .WORD   0       ;;USER DEFINED
001210  000000                  $TMP5:  .WORD   0       ;;USER DEFINED
001212  000000                  $TIMES: 0               ;;MAX. NUMBER OF ITERATIONS
001214  000000                  $ESCAPE:0               ;;ESCAPE ON ERROR ADDRESS
001216  177607  000377      $BELL:  .ASCIZ  <207><377><377> ;;CODE FOR BELL
001222  077                     $QUES:  .ASCII  /?/     ;;QUESTION MARK
001223  015                     $CRLF:  .ASCII  <15>    ;;CARRIAGE RETURN
```

```
733  001224  000012                    $LF:    .ASCIZ  <12>            ;;LINE FEED
734                                     ;;*****************************************************************
735                                     .SBTTL  APT MAILBOX-ETABLE
736
737                                     ;;*****************************************************************
738                                     .EVEN
739  001226                             $MAIL:                          ;;APT MAILBOX
740  001226  000000                     $MSGTY: .WORD   AMSGTY          ;;MESSAGE TYPE CODE
741  001230  000000                     $FATAL: .WORD   AFATAL          ;;FATAL ERROR NUMBER
742  001232  000000                     $TESTN: .WORD   ATESTN          ;;TEST NUMBER
743  001234  000000                     $PASS:  .WORD   APASS           ;;PASS COUNT
744  001236  000000                     $DEVCT: .WORD   ADEVCT          ;;DEVICE COUNT
745  001240  000000                     $UNIT:  .WORD   AUNIT           ;;I/O UNIT NUMBER
746  001242  000000                     $MSGAD: .WORD   AMSGAD          ;;MESSAGE ADDRESS
747  001244  000000                     $MSGLG: .WORD   AMSGLG          ;;MESSAGE LENGTH
748  001246                             $ETABLE:                        ;;APT ENVIRONMENT TABLE
749  001246      000                    $ENV:   .BYTE   AENV            ;;ENVIRONMENT BYTE
750  001247      000                    $ENVM:  .BYTE   AENVM           ;;ENVIRONMENT MODE BITS
751  001250  000000                     $SWREG: .WORD   ASWREG          ;;APT SWITCH REGISTER
752  001252  000000                     $USWR:  .WORD   AUSWR           ;;USER SWITCHES
753  001254  000000                     $CPUOP: .WORD   ACPUOP          ;;CPU TYPE,OPTIONS
754                                     ;*                              BITS 15-11=CPU TYPE
755                                     ;*                                      11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
756                                     ;*                                      11/70=06,PDQ=07,Q=10
757                                     ;*                              BIT 10=REAL TIME CLOCK
758                                     ;*                              BIT  9=FLOATING POINT PROCESSOR
759                                     ;*                              BIT  8=MEMORY MANAGEMENT
760  001256      000                    $MAMS1: .BYTE   AMAMS1          ;;HIGH ADDRESS,M.S. BYTE
761  001257      000                    $MTYP1: .BYTE   AMTYP1          ;;MEM. TYPE,BLK#1
762                                     ;*                              MEM.TYPE BYTE   --  (HIGH BYTE)
763                                     ;*                                      900 NSEC CORE=001
764                                     ;*                                      300 NSEC BIPOLAR=002
765                                     ;*                                      500 NSEC MOS=003
766  001260  000000                     $MADR1: .WORD   AMADR1          ;;HIGH ADDRESS,BLK#1
767                                     ;*                                MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
768  001262      000                    $MAMS2: .BYTE   AMAMS2          ;;HIGH ADDRESS,M.S. BYTE
769  001263      000                    $MTYP2: .BYTE   AMTYP2          ;;MEM.TYPE,BLK#2
770  001264  000000                     $MADR2: .WORD   AMADR2          ;;MEM.LAST ADDRESS,BLK#2
771  001266      000                    $MAMS3: .BYTE   AMAMS3          ;;HIGH ADDRESS,M.S.BYTE
772  001267      000                    $MTYP3: .BYTE   AMTYP3          ;;MEM.TYPE,BLK#3
773  001270  000000                     $MADR3: .WORD   AMADR3          ;;MEM.LAST ADDRESS,BLK#3
774  001272      000                    $MAMS4: .BYTE   AMAMS4          ;;HIGH ADDRESS,M.S.BYTE
775  001273      000                    $MTYP4: .BYTE   AMTYP4          ;;MEM.TYPE,BLK#4
776  001274  000000                     $MADR4: .WORD   AMADR4          ;;MEM.LAST ADDRESS,BLK#4
777  001276  000000                     $VECT1: .WORD   AVECT1          ;;INTERRUPT VECTOR#1,BUS PRIORITY#1
778  001300  000000                     $VECT2: .WORD   AVECT2          ;;INTERRUPT VECTOR#2BUS PRIORITY#2
779  001302  000000                     $BASE:  .WORD   ABASE           ;;BASE ADDRESS OF EQUIPMENT UNDER TEST
780  001304  000000                     $DEVM:  .WORD   ADEVM           ;;DEVICE MAP
781  001306  000000                     $CDW1:  .WORD   ACDW1           ;;CONTROLLER DESCRIPTION WORD#1
782  001310  000000                     $CDW2:  .WORD   ACDW2           ;;CONTROLLER DESCRIPTION WORD#2
783  001312  000000                     $DDW0:  .WORD   ADDW0           ;;DEVICE DESCRIPTOR WORD#0
784  001314  000000                     $DDW1:  .WORD   ADDW1           ;;DEVICE DESCRIPTOR WORD#1
785  001316  000000                     $DDW2:  .WORD   ADDW2           ;;DEVICE DESCRIPTOR WORD#2
786  001320  000000                     $DDW3:  .WORD   ADDW3           ;;DEVICE DESCRIPTOR WORD#3
787  001322  000000                     $DDW4:  .WORD   ADDW4           ;;DEVICE DESCRIPTOR WORD#4
788  001324  000000                     $DDW5:  .WORD   ADDW5           ;;DEVICE DESCRIPTOR WORD#5
```

```
789  001326  000000                  $DDW6:  .WORD   ADDW6   ;;DEVICE DESCRIPTOR WORD#6
790  001330  000000                  $DDW7:  .WORD   ADDW7   ;;DEVICE DESCRIPTOR WORD#7
791  001332  000000                  $DDW8:  .WORD   ADDW8   ;;DEVICE DESCRIPTOR WORD#8
792  001334  000000                  $DDW9:  .WORD   ADDW9   ;;DEVICE DESCRIPTOR WORD#9
793  001336  000000                  $DDW10: .WORD   ADDW10  ;;DEVICE DESCRIPTOR WORD#10
794  001340  000000                  $DDW11: .WORD   ADDW11  ;;DEVICE DESCRIPTOR WORD#11
795  001342  000000                  $DDW12: .WORD   ADDW12  ;;DEVICE DESCRIPTOR WORD#12
796  001344  000000                  $DDW13: .WORD   ADDW13  ;;DEVICE DESCRIPTOR WORD#13
797  001346  000000                  $DDW14: .WORD   ADDW14  ;;DEVICE DESCRIPTOR WORD#14
798  001350  000000                  $DDW15: .WORD   ADDW15  ;;DEVICE DESCRIPTOR WORD#15
799
800
801  001352                          $ETEND:
802
803
804  001352  177746                  CONTRL: .WORD   177746          ;CACHE CONTROL REGISTER
805  001354  177766                  CPUERR: .WORD   177766          ;CPU ERROR REGISTER HOLDS
806                                                                  ;CONDITION THAT CAUSED THE TRAP
807                                                                  ;TO ERRVEC(000004)
808  001356  177752                  HITMIS: .WORD   177752          ;HIT MISS REGISTER "1"
809                                                                  ;IMPLIES HIT IN CACHE
810  001360  177744                  MEMERR: .WORD   177744          ;CACHE ERROR REGISTER
811  001362  000000                  FSTTST: .WORD   0               ;HOLDS THE INDEX TO
812                                                                  ;THE STARTING ADDRESS TABLE
813  001364  000000                  CPUEXP: .WORD   0               ;HOLDS EXPECTED CPU ERROR CONDITION
814  001366  000000                  MMEXP:  .WORD                   ;HOLDS EXPECTED MEMORY MANAGEMENT ABORT CONDITION
815  001370  000000                  STKEXP: .WORD   0               ;HOLDS TYPE OF STACK VIOLATION EXPECTED
816  001372  000000                  PADRSL: .WORD   0               ;HOLDS THE LOWER 16 BITS OF A 18-BIT
817                                                                  ;ADDRESS GENERATED FOR TYPE OUT
818  001374  000000                  PADRSH: .WORD   0               ;HOLDS THE UPPER 6 BITS OF A 18-BIT
819                                                                  ;ADDRESS GENERATED FOR TYPE OUT
820  001376  000000                  PPARER: .WORD   0               ;HOLDS PARITY ERROR REG
821  001400  000000                  PCONTR: .WORD   0               ;HOLDS CACHE CONTROL REG
822  001402  000000                  PHITMI: .WORD   0               ;HOLDS CACHE HIT MISS REG
823  001404  000000                  PMMR0:  .WORD   0               ;HOLDS MEMORY MANAGEMENT REGISTER 0
824  001406  000000                  PMMR2:  .WORD   0               ;HOLDS MEMORY MANAGEMENT REGISTER 2
825  001410  000000                  PCPUER: .WORD   0               ;HOLDS CPU ERROR REGISTER.
826  001412  000000                  BADPC:  .WORD   0               ;HOLDS PC AT ABORT OR TRAP TIME.
827  001414  000000                  TESTNO: .WORD   0               ;HOLDS TEST NUMBER OF LAST ERROR.
828  001416  000000                  DATAOR: .WORD   0               ;HOLDS LOGICAL OR OF BAD DATA
829  001420  000000                  DATAND: .WORD   0               ;HOLDS LOGICAL AND OF BAD DATA
830  001422  000000                  PATTOR: .WORD   0               ;HOLDS LOGICAL OR OF PATTERN LOADED
831  001424  000000                  PATAND: .WORD   0               ;HOLDS LOGICAL AND OF PATTERN LOADED
832  001426  000000                  ADDROR: .WORD   0               ;HOLDS LOGICAL OR OF ADDRESS
833  001430  000000                  ADRAND: .WORD   0               ;HOLDS LOGICAL AND OF ADDRESS
834  001432  000000                  ERRCNT: .WORD   0               ;HOLDS NUMBER OF ERRORS ON TEST
835  001434  000000                  HOLFLG: .WORD   0               ;HOLDS NUMBER OF CONSECUTIVE TIME OUTS
836                                                                  ;OCCURRING IN A HOLE IN MEMORY
837  001436  000000                  OLDPC:  .WORD   0               ;HOLDS THE RETURN ADDRESS
838                                                                  ;IN CASE OF A LOOP ON ERROR
839  001440  000000                  OLDPS:  .WORD   0               ;HOLDS THE OLD PROCESSOR STATUS
840                                                                  ;IN CASE OF A LOOP ON ERROR
841  001442  000340                  OLDPSW: .WORD   000340          ;HOLDS OLD PSW SO THAT THE T-BIT
842                                                                  ;CAN BE RESTORED PROPERLY
843  001444  000000                  RETRY:  .WORD   0               ;FLAG TO DECIDE IF ONE PARITY HAS
844                                                                  ;HAS BEEN ATTEMPTED
```

```
  845  001446  000000                NXTTST: .WORD   0               ;HOLDS ADDRESS OF THE NEXT TEST
  846  001450     000                FORTY:  .BYTE   0               ;PROCESSOR INDICATOR
  847                                                                ;IF = 0   RUNNING ON AN 11/6X
  848                                                                ;IF = 1   RUNNING ON AN 11/40
  849          001452                        .EVEN
  850
  851                                ;;  THIS AREA STARTS THE DATA TABLE WHERE ANY TABLE REFERENCE WILL
  852                                ;;  REFER TO.  ALL DATA WORDS WILL BE LOADED HERE IN ONE SPOT SO THAT
  853                                ;;  IT WILL BE EASIER FOR YOU TO FIND OUT WHAT THAT WORD IS.
  854
  855
  856
  857  001452                        PARTAB:                         ;THIS IS THE TABLE OF THE FIRST
  858                                                                ;PAR OR PDR OF EACH GROUP.  THEY
  859                                                                ;WILL BE USED FOR A DUAL ADDRESSING
  860                                                                ;TEST BETWEEN GROUPS.
  861  001452  172300                        .WORD   172300          ;KIPDR0
  862  001454  172340                        .WORD   172340          ;KIPAR0
  863  001456  177600                        .WORD   177600          ;UIPDR0
  864  001460  177640                        .WORD   177640          ;UIPAR0
  865  001462  020776                STRTAB: .WORD   TST1            ;STARTING ADDRESS OF TEST ONE
  866  001464  021510                        .WORD   ENTPT2          ;ADDRESS OF ENTRY POINT 2
  867  001466  022072                        .WORD   ENTPT3          ;ADDRESS OF ENTRY POINT 3
  868  001470  025066                        .WORD   ENTPT4          ;ADDRESS OF ENTRY POINT 4
  869  001472  027204                        .WORD   ENTPT5          ;ADDRESS OF ENTRY POINT 5
  870  001474  033674                        .WORD   ENTPT6          ;ADDRESS OF ENTRY POINT 6
  871  001476  036000                        .WORD   ENTPT7          ;ADDRESS OF ENTRY POINT 7
```

```
                                        ERROR POINTER TABLE

                                        .SBTTL  ERROR POINTER TABLE

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

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

001500                                  $ERRTB:

                                        ;ITEM 1
001500  002520                          EM1                     ;UNEXPECTED CPU TRAP OR ABORT THRU 'ERRVEC' (004)
001502  007755                          DH1                     ;TESTNO  PC AT ABORT
001504  013726                          DT1                     ;TESTNO,BADPC,0
001506  014660                          DF1                     ;0,0

                                        ;ITEM 2
001510  002520                          EM1                     ;UNEXPECTED CPU TRAP OR ABORT THRU 'ERRVEC' (004)
001512  010000                          DH2                     ;CPU ERR TESTNO  PC AT ABORT
001514  013734                          DT2                     ;PCPUER,TESTNO,BADPC,0
001516  014662                          DF2                     ;0,0,0

                                        ;ITEM 3
001520  002601                          EM3                     ;UNEXPECTED CACHE PARITY ERROR THRU 'CACHVEC' (114)
001522  010034                          DH3                     ;MEM ERR CONTROL HITMISS
                                                                ;REGISTR REGISTR REGISTR TESTNO  PC AT ABORT
001524  013744                          DT3                     ;PPARER,PCONTR,PHITMI,TESTNO,BADPC,0
001526  014665                          DF3                     ;0,0,0,0,0

                                        ;ITEM 4
001530  002717                          EM4                     ;UNEXPECTED MAIN MEMORY PARITY ERROR THRU 'CACHVEC' (114)
001532  010034                          DH3                     ;MEM ERR  CONTROL HITMISS
                                                                ;REGISTR REGISTR REGISTR TESTNO  PC AT ABORT
001534  013744                          DT3                     ;PPARER,PCONTR,PHITMI,TESTNO,BADPC,0
001536  014665                          DF3                     ;0,0,0,0,0

                                        ;ITEM 5
001540  003042                          EM5                     ;UNEXPECTED MEMORY MANAGEMENT TRAP OR ABORT
001542  010142                          DH5                     ;ERROR   VIRTUAL
                                                                ;REGISTR ADDRESS TESTNO  PC AT ABORT
001544  013760                          DT5                     ;PMMR0,PMMR2,TESTNO,BADPC,0
001546  014672                          DF5                     ;0,0,0,0

                                        ;ITEM 6
001550  003115                          EM6                     ;MEMORY MANAGEMENT TRAP OR ABORT HAD INCORRECT CONDITION
001552  010226                          DH6                     ;EXPECTD ERROR   VIRTUAL
                                                                ;CONDITN REGISTR ADDRESS TESTNO  PC AT ABORT
001554  013772                          DT6                     ;MMEXP,PMMR0,PMMR2,TESTNO,BADPC,0
001556  014676                          DF6                     ;0,0,0,0,0
```

```
                                        ;ITEM 7
001560  003205                          EM7             ;MEMORY MANAGEMENT REGISTER 0 WILL NOT CLEAR
001562  010332                          DH7             ;(MMR0)  TESTNO  ERRORPC
001564  014006                          DT7             ;$REG1,TESTNO,$ERRPC,0
001566  014703                          DF7             ;0,0,0
                                        ;ITEM 10
001570  003261                          EM10            ;CAN'T SET 160000 INTO MMR0
001572  010332                          DH7             ;(MMR0)  TESTNO  ERRORPC
001574  014006                          DT7             ;$REG1,TESTNO,$ERRPC,0
001576  014703                          DF7             ;0,0,0
                                        ;ITEM 11
001600  003312                          EM11            ;GOT THE WRONG DATA BACK FROM MMR0
001602  010362                          DH11            ;LOADED  RECEIVD TESTNO  ERRORPC
001604  014016                          DT11            ;$REG2,$REG1,TESTNO,$ERRPC,0
001606  014706                          DF11            ;0,0,0,0
                                        ;ITEM 12
001610  003354                          EM12            ;MMR2 DID NOT TRACK PROPERLY
001612  010422                          DH12            ;EXPECTD (MMR2)  TESTNO  ERRORPC
001614  014030                          DT12            ;$REG3,$REG2,TESTNO,$ERRPC
001616  014712                          DF12            ;0,0,0,0
                                        ;ITEM 13
001620  003410                          EM13            ;SUMMARY OF PAR/PDR REFERENCE TIMEOUTS
001622  010462                          DH13            ;ADDROR  ADDRAND TESTNO ERRORPC  #ERRORS
001624  014042                          DT13            ;ADDROR,ADRAND,TESTNO,$ERRPC,$TMP5,0
001626  014716                          DF13            ;0,0,0,0,1
                                        ;ITEM 14
001630  003456                          EM14            ;FOLLOWING PAR/PDR WILL NOT ZERO
001632  010533                          DH14            ;ADDRESS DATA    TESTNO  ERRORPC
001634  014056                          DT14            ;$REG0,$REG2,TESTNO,$ERRPC,0
001636  014723                          DF14            ;0,0,0,0
                                        ;ITEM 15
001640  003517                          EM15            ;SUMMARY OF DUAL ADDRESSING ERRORS
001642  010573                          DH15            ;ADDROR  ADDRAND ADDROR  ADDRAND
                                                        ;LOADED  LOADED ENABLED ENABLED TESTNO ERRORPC  #ERRORS
001644  014070                          DT15            ;ADDROR,ADRAND,DATAOR,DATAND,TESTNO,$ERRPC,$TMP5,0
001646  014727                          DF15            ;0,0,0,0,0,0,1
                                        ;ITEM 16
001650  003561                          EM16            ;SUMMARY OF COUNT PATTERN FAILURES
001652  010722                          DH16            ;ADDROR  ADDRAND PATRNOR PATRNAD DATAOR  DATAAND TESTNO ERRORPC
001654  014110                          DT16            ;ADDROR ADRAND,PATTOR,PATAMD,DATAOR,DATAND,TESTNO,$ERRPC,$TMP5,0
001656  014736                          DF16            ;0,0,0,0,0,0,0,0,1
                                        ;ITEM 17
001660  003623                          EM17            ;ERROR IN BYTE ADDRESSING OF PAR/PDR
001662  011031                          DH17            ;ADDRESS EXPECTD RECEIVD TESTNO  ERRORPC
001664  014134                          DT17            ;$REG0,$REG2,$REG1,TESTNO,$ERRPC,0
001666  014747                          DF17            ;0,0,0,0,0
```

ERROR POINTER TABLE

```
                                        ;ITEM 20
001670  003667                          EM20            ;ONE OF THE REGISTERS TIMED OUT
001672  011101                          DH20            ;REGADDR TESTNO  ERRORPC
001674  014150                          DT20            ;$REG0,TESTNO,$ERRPC,0
001676  014754                          DF20            ;0,0,0
                                        ;ITEM 21
001700  003726                          EM21            ;STACK LIMIT REGISTER WOULD NOT CLEAR
001702  011131                          DH21            ;REGADDR DATA    TESTNO  ERRORPC
001704  014160                          DT21            ;$REG0,$REG1,TESTNO,$ERRPC,0
001706  014757                          DF21            ;0,0,0,0
                                        ;ITEM 22
001710  003773                          EM22            ;ERROR LOG REG.DID NOT HAVE CORRECT BIT SET
001712  011171                          DH22            ;REGADR EXPECTD RECEIVD TESTNO ERRORPC
001714  014172                          DT22            ;$TMP4,$TMP5,$REG0,TESTNO,$ERRPC,0
001716  014763                          DF22            ;0,0,0,0,0
                                        ;ITEM 23
001720  004047                          EM23            ;LOWER BYTE OF P.S.W. NOT CORRECT
001722  011240                          DH23            ;REGADDR DATAREC DATAEXP TESTNO  ERRORPC
001724  014206                          DT23            ;$REG0,$REG1,$REG2,TESTNO,$ERRPC,0
001726  014770                          DF23            ;0,0,0,0,0
                                        ;ITEM 24
001730  004110                          EM24            ;MMR1 DID NOT READ ALL ZEROES
001732  011131                          DH21            ;REGADDR DATA    TESTNO  ERRORPC
001734  014160                          DT21            ;$REG0,$REG1,TESTNO,$ERRPC,0
001736  014757                          DF21            ;0,0,0,0
                                        ;ITEM 25
001740  004145                          EM25            ;WRONG DATA BACK FROM STACK LIMIT REGISTER
001742  011240                          DH23            ;REGADDR DATAREC DATAEXP TESTNO  ERRORPC
001744  014206                          DT23            ;$REG0,$REG1,$REG2,TESTNO,$ERRPC,0
001746  014770                          DF23            ;0,0,0,0,0
                                        ;ITEM 26
001750  004217                          EM26            ;LOADED MORE THAN UPPER BYTE OF STACK LIMIT REGISTER
001752  011240                          DH23            ;REGADDR DATAREC DATAEXP TESTNO  ERRORPC
001754  014206                          DT23            ;$REG0,$REG1,$REG2,TESTNO,$ERRPC,0
001756  014770                          DF23            ;0,0,0,0,0
                                        ;ITEM 27
001760  004303                          EM27            ;KERNEL STACK POINTER NOT 1100 AFTER LOADING USP
001762  011310                          DH27            ;KSP     TESTNO  ERRORPC
001764  014222                          DT27            ;$REG0,TESTNO,$ERRPC,0
001766  014775                          DF27            ;0,0,0
                                        ;ITEM 30
001770  004363                          EM30            ;DUAL ADDRESSING BETWEEN PAR/PDR GROUPS
001772  011340                          DH30            ;INDEX   INDEX   PAR/PDR
                                                        ;EXPECTD RECEIVD ADRREAD TESTNO  ERRORPC
001774  014232                          DT30            ;$REG0,$REG1,$REG2,TESTNO,$ERRPC,0
001776  015000                          DF30            ;0,0,0,0,0
                                        ;ITEM 31
```

```
1040  002000  004432                EM31                ;BAD RELOCATION, ON STORING DATA 18-BIT MAPPING
1041  002002  011440                DH31                ;VIRTUAL          PHYSCAL DATA    DATA
1042                                                     ;ADDRESS KIPAR4  ADDRESS EXPECTD RECEIVD TESTNO ERRORPC
1043  002004  014246                DT31                ;$REG0,$TMP1,$TMP2,$REG1,$REG2,TESTNO,$ERRPC,0
1044  002006  015005                DF31                ;0,0,0,0,0,0,0,0
1045
1046                                ;ITEM 32
1047  002010  004511                EM32                ;18-BIT MAPPING POSSIBLE HOLE IN MAIN MEMORY FROM
1048  002012  011574                DH32                ;STRTADR FNSHADR TESTNO ERRORPC
1049  002014  014266                DT32                ;$TMP1,$TMP2,TESTNO,$ERRPC,0
1050  002016  015014                DF32                ;2,2,0,0
1051
1052                                ;ITEM 33
1053  002020  004572                EM33                ;FAULTY CARRY PROPAGATION 18-BIT MAPPING.
1054  002022  011634                DH33                ;PATTERN DATA     ADDRESS
1055                                                     ;LOADED  FETCHED INTENDD  KIPAR4  KIPAR5  TESTNO  ERRORPC
1056  002024  014300                DT33                ;$REG2,$REG3,$REG0,$TMP3,$TMP4,TESTNO,$ERRPC,0
1057  002026  015020                DF33                ;0,0,0,0,0,0,0,0
1058
1059                                ;ITEM 34
1060  002030  004643                EM34                ;DIDN'T GET WRAP AROUND TO ADDRESS ZERO
1061  002032  011754                DH34                ;DATA    TESTNO  ERRORPC
1062  002034  014320                DT34                ;$REG1,TESTNO,$ERRPC,0
1063  002036  015027                DF34                ;0,0,0
1064
1065                                ;ITEM 35
1066  002040  004712                EM35                ;PAGE LENGTH ABORT AT WRONG VIRTUAL ADDRESS
1067  002042  012004                DH35                ;PGLENFD VABLKNO TESTNO  ERRORPC
1068  002044  014330                DT35                ;$REG1,$REG3,TESTNO,$ERRPC,0
1069  002046  015032                DF35                ;0,0,0,0
1070
1071                                ;ITEM 36
1072  002050  004765                EM36                ;NO PAGE LENGTH ABORT, WHEN CONDITION CORRECT
1073  002052  012004                DH35                ;PGLENFD VABLKNO TESTNO  ERRORPC
1074  002054  014330                DT35                ;$REG1,$REG3,TESTNO,$ERRPC,0
1075  002056  015032                DF35                ;0,0,0,0
1076
1077                                ;ITEM 37
1078  002060  005042                EM37                ;DID NOT ABORT ON ACCESS OF NON-RESIDENT PAGE THRU KIPDR5
1079  002062  012044                DH37                ;KIPDR5  KIPAR5  VIRTADR TESTNO  ERRORPC
1080  002064  014342                DT37                ;$TMP0,$TMP1,$TMP2,TESTNO,$ERRPC,0
1081  002066  015036                DF37                ;0,0,0,0,0
1082
1083                                ;ITEM 40
1084  002070  005133                EM40                ;DID NOT ABORT ON WRITE TO READ ONLY PAGE THRU KIPDR4
1085  002072  012114                DH40                ;KIPDR4  KIPAR4  VIRTADR TESTNO  ERRORPC
1086  002074  014342                DT37                ;$TMP0,$TMP1,$TMP2,TESTNO,$ERRPC,0
1087  002076  015036                DF37                ;0,0,0,0,0
1088
1089                                ;ITEM 41
1090  002100  005220                EM41                ;POINTER VALUE 400 CAUSED STACK VIOLATION
1091  002102  012164                DH41                ;STKPTR STKLMT TESTNO ERRORPC
1092  002104  014356                DT41                ;$REG2,$REG3,TESTNO,$ERRPC,0
1093  002106  015043                DF41                ;0,0,0,0
1094
1095                                ;ITEM 42
```

```
1096  002110  005271                  EM42                    ;YELLOW ZONE VIOLATION EXPECTED-GOT NONE
1097  002112  012164                  DH41                    ;STKPTR STKLMT TESTNO ERRORPC
1098  002114  014356                  DT41                    ;$REG2,$REG3,TESTNO,$ERRPC,0
1099  002116  015043                  DF41                    ;0,0,0,0
1100
1101                                  ;ITEM 43
1102  002120  005341                  EM43                    ;YELLOW ZONE VIOLATION EXPECTED-GOT RED
1103  002122  012164                  DH41                    ;STKPTR STKLMT TESTNO ERRORPC
1104  002124  014356                  DT41                    ;$REG2,$REG3,TESTNO,$ERRPC,0
1105  002126  015043                  DF41                    ;0,0,0,0
1106
1107                                  ;ITEM 44
1108  002130  005410                  EM44                    ;RED ZONE VIOLATION EXPECTED-GOT NONE
1109  002132  012164                  DH41                    ;STKPTR STKLMT TESTNO ERRORPC
1110  002134  014356                  DT41                    ;$REG2,$REG3,TESTNO,$ERRPC,0
1111  002136  015043                  DF41                    ;0,0,0,0
1112
1113                                  ;ITEM 45
1114  002140  005455                  EM45                    ;RED ZONE VIOLATION DID NOT CAUSE ABORT
1115  002142  012164                  DH41                    ;STKPTR STKLMT TESTNO ERRORPC
1116  002144  014356                  DT41                    ;$REG2,$REG3,TESTNO,$ERRPC,0
1117  002146  015043                  DF41                    ;0,0,0,0
1118
1119                                  ;ITEM 46
1120  002150  005524                  EM46                    ;RED ZONE VIOLATION EXPECTED - GOT YELLOW
1121  002152  012164                  DH41                    ;STKPTR STKLMT TESTNO ERRORPC
1122  002154  014356                  DT41                    ;$REG2,$REG3,TESTNO,$ERRPC,0
1123  002156  015043                  DF41                    ;0,0,0,0
1124
1125                                  ;ITEM 47
1126  002160  005575                  EM47                    ;ERROR DURING ERROR-ON-ERROR TRAP SEQUENCE
1127                                                          ;EXPECTED:
1128  002162  012224                  DH47                    ;PSW    STK PTR (MMR0) (MMR2) TESTNO ERRORPC
1129                                                          ;140017 -11/6X- 000000 020151
1130                                                          ;140000 -11/40- 4&1074
1131                                                          ;RECEIVED:
1132                                                          ;PSW    PC      STK PTR (MMR0) (MMR2) TESTNO ERRORPC
1133  002164  014370                  DT47                    ;$REG2,$REG3,$REG1,PMMR0,PMMR2,TESTNO,$ERRPC,0
1134  002166  015047                  DF47                    ;0,0,0,0,0,0,0
1135
1136                                  ;ITEM 50
1137  002170  005661                  EM50                    ;AT LEAST ONE M.M. STATUS REGISTERS WAS CLOCKED AFTER BEING LOCK
1138  002172  012467                  DH50                    ;ORIGINAL DATA          NEW DATA
1139                                                          ;(MMR0)  (MMR2)  (MMR0)  (MMR2)  TESTNO  ERRORPC
1140  002174  014410                  DT50                    ;PMMR0,PMMR2,$TMP0,$TMP2,TESTNO,$ERRPC,0
1141  002176  015056                  DF50                    ;0,0,0,0,0,0
1142
1143                                  ;ITEM 51
1144  002200  005763                  EM51                    ;DID NOT CHANGE MAPPING TO USER MODE
1145  002202  012044                  DH37                    ;TESTNO  ERRORPC
1146  002204  014342                  DT37                    ;TESTNO,$ERRPC,0
1147  002206  015036                  DF37                    ;0,0
1148
1149                                  ;ITEM 52
1150  002210  006027                  EM52                    ;ABORT CONDITION INCORRECT EXPECTING 100153
1151  002212  012600                  DH52                    ;RECEIVD (MMR2)  TESTNO  ERRORPC
```

```
1152  002214  014426                  DT52                    ;PMMR0,PMMR2,TESTNO,$ERRPC,0
1153  002216  015064                  DF52                    ;0,0,0,0
1154
1155                                  ;ITEM 53
1156  002220  006102                  EM53                    ;MMR2 LOCKED UP THE WRONG VIRTUAL ADDRESS
1157  002222  012640                  DH53                    ;VIRTADR (MMR2)  TESTNO  ERRORPC
1158  002224  014440                  DT53                    ;$REG1,PMMR2,TESTNO,$ERRPC,0
1159  002226  015070                  DF53                    ;0,0,0,0
1160
1161                                  ;ITEM 54
1162  002230  006153                  EM54                    ;MMR0 LOCKED UP THE WRONG PAGE NUMBER
1163  002232  012700                  DH54                    ;EXPECTD (MMR0)  TESTNO  ERRORPC
1164  002234  014452                  DT54                    ;$REG2,PMMR0,TESTNO,$ERRPC,0
1165  002236  015074                  DF54                    ;0,0,0,0
1166
1167                                  ;ITEM 55
1168  002240  006220                  EM55                    ;W-BIT NOT SET ON WRITE TO PAGE 4
1169  002242  012740                  DH55                    ;KIPDR4  TESTNO  ERRORPC
1170  002244  014464                  DT55                    ;$TMP0,TESTNO,$ERRPC,0
1171  002246  015100                  DF55                    ;0,0,0
1172
1173                                  ;ITEM 56
1174  002250  006261                  EM56                    ;W-BIT DID NOT REMAIN SET ON M.M. ABORT AT PAGE 4
1175  002252  012740                  DH55                    ;KIPDR4  TESTNO  ERRORPC
1176  002254  014464                  DT55                    ;$TMP0,TESTNO,$ERRPC,0
1177  002256  015100                  DF55                    ;0,0,0
1178
1179                                  ;ITEM 57
1180  002260  006342                  EM57                    ;W-BIT DID NOT CLEAR ON WRITE TO KIPDR4
1181  002262  012740                  DH55                    ;KIPDR4  TESTNO  ERRORPC
1182  002264  014464                  DT55                    ;$TMP0,TESTNO,$ERRPC,0
1183  002266  015100                  DF55                    ;0,0,0
1184
1185                                  ;ITEM 60
1186  002270  006411                  EM60                    ;W-BIT WRONG ON WRITE TO PSW
1187  002272  012770                  DH60                    ;KIPDR7  TESTNO  ERRORPC
1188  002274  014464                  DT55                    ;$TMP0,TESTNO,$ERRPC,0
1189  002276  015100                  DF55                    ;0,0,0
1190
1191                                  ;ITEM 61
1192  002300  006445                  EM61                    ;W-BIT SET AFTER WRITE TO PDR 4
1193  002302  012770                  DH60                    ;KIPDR7  TESTNO  ERRORPC
1194  002304  014464                  DT55                    ;$TMP0,TESTNO,$ERRPC,0
1195  002306  015100                  DF55                    ;0,0,0
1196
1197                                  ;ITEM 62
1198  002310  006504                  EM62                    ;DUAL MAPPING BETWEEN PAGES
1199  002312  013020                  DH62                    ;GDPAGE  BDPAGE  TESTNO  ERRORPC
1200  002314  014474                  DT62                    ;$REG3,$REG1,TESTNO,$ERRPC,0
1201  002316  015103                  DF62                    ;0,0,0,0
1202
1203                                  ;ITEM 63
1204  002320  006537                  EM63                    ;NO PAGE HAD ITS BIT SET
1205  002322  013060                  DH63                    ;TSTPAGE CONTENT TESTNO  ERRORPC
1206  002324  014506                  DT63                    ;$REG3,$REG0,TESTNO,$ERRPC,0
1207  002326  015107                  DF63                    ;0,0,0,0
```

```
1208
1209                                    ;ITEM 64
1210  002330  006567                    EM64                    ;DID NOT PICK UP CORRECT STACK POINTER
1211  002332  013120                    DH64                    ;EXPECTD RECEIVD TESTNO  ERRORPC
1212  002334  014520                    DT64                    ;$REG2,$REG1,TESTNO,$ERRPC,0
1213  002336  015113                    DF64                    ;0,0,0,0
1214
1215                                    ;ITEM 65
1216  002340  006635                    EM65                    ;CURRENT MODE STACK NOT PUSHED IN MFP INSTRUCTION
1217  002342  012044                    DH37                    ;TESTNO  ERRORPC
1218  002344  014342                    DT37                    ;TESTNO,$ERRPC,0
1219  002346  015036                    DF37                    ;0,0
1220
1221                                    ;ITEM 66
1222  002350  006716                    EM66                    ;WRONG DATA FETCHED BY MFP INSTRUCTION
1223  002352  013160                    DH66                    ;EXPECTD RECEIVD TESTNO  ERRORPC
1224  002354  014532                    DT66                    ;$REG0,$REG1,TESTNO,$ERRPC,0
1225  002356  015117                    DF66                    ;0,0,0,0
1226
1227                                    ;ITEM 67
1228  002360  006764                    EM67                    ;TRIED TO REFERENCE NON-RESIDENT PAGE
1229  002362  013220                    DH67                    ;(MMR0)  (MMR2)  TESTNO  ERRORPC
1230  002364  014544                    DT67                    ;PMMR0,PMMR2,TESTNO,$ERRPC,0
1231  002366  015123                    DF67                    ;0,0,0,0
1232
1233                                    ;ITEM 70
1234  002370  007031                    EM70                    ;STACK POINTER NOT CHANGED BY MTP INSTRUCTION
1235  002372  013260                    DH70                    ;STKPTR  TESTNO  ERRORPC
1236  002374  014556                    DT70                    ;$REG1,TESTNO,$ERRPC,0
1237  002376  015127                    DF70                    ;0,0,0
1238
1239                                    ;ITEM 71
1240  002400  007106                    EM71                    ;INCORRECT STORE BY MTP INSTRUCTION
1241  002402  013310                    DH71                    ;GDDATA  STORED  TESTNO  ERRORPC
1242  002404  014532                    DT66                    ;$REG0,$REG1,TESTNO,$ERRPC,0
1243  002406  015117                    DF66                    ;0,0,0,0
1244
1245                                    ;ITEM 72
1246  002410  007151                    EM72                    ;ABORTED THRU RIGHT VECTOR BUT PICKED UP WRONG PSW
1247  002412  013350                    DH72                    ;(PSW)   TESTNO  ERRORPC EXPECTING XXX340
1248  002414  014566                    DT72                    ;$REG0,TESTNO,$ERRPC,0
1249  002416  015132                    DF72                    ;0,0,0
1250
1251                                    ;ITEM 73
1252  002420  007233                    EM73                    ;ABORTED THRU USER SPACE, USER PSW IS XXX000
1253  002422  013421                    DH73                    ;(PSW)   TESTNO  ERRORPC
1254  002424  014566                    DT72                    ;$REG0,TESTNO,$ERRPC,0
1255  002426  015132                    DF72                    ;0,0,0
1256
1257                                    ;ITEM 74
1258  002430  007307                    EM74                    ;DID NOT ABORT REFERENCE TO A MEMORY MANAGEMENT REGISTER
1259  002432  013451                    DH74                    ;TESTNO  ERRORPC
1260  002434  014576                    DT74                    ;TESTNO,$ERRPC,0
1261  002436  015135                    DF74                    ;0,0
1262
1263                                    ;ITEM 75
```

```
                                ERROR POINTER TABLE

1264   002440  007377           EM75            ;CPU ERROR REG. DID NOT REPORT YELLOW ZONE
1265   002442  013470           DH75            ;STKPTR STKLMT CPUERR TESTNO ERRORPC
1266   002444  014604           DT75            ;$REG2,$REG3,PCPUER,TESTNO,$ERRPC,0
1267   002446  015137           DF75            ;0,0,0,0,0
1268
1269                            ;ITEM 76
1270   002450  007451           EM76            ;CPU ERROR REG. DID NOT REPORT RED ZONE
1271   002452  013470           DH75            ;STKPTR STKLMT CPUERR TESTNO ERRORPC
1272   002454  014604           DT75            ;$REG2,$REG3,PCPUER,TESTNO,$ERRPC,0
1273   002456  015137           DF75            ;0,0,0,0,0
1274
1275                            ;ITEM 77
1276   002460  002717           EM4             ;UNEXPECTED MAIN MEMORY PARITY ERROR THRU 'CACHVEC' (114)
1277   002462  007755           DH1             ;TESTNO  PC AT ABORT
1278   002464  013726           DT1             ;TESTNO,BADPC,0
1279   002466  014660           DF1             ;0,0
1280
1281
1282
1283   002470                   ER200:          ;STARTING ADDRESS FOR ITEM 201-377
1284
1285                            ;ITEM 201
1286   002470  007520           EM201           ;THE FOLLOWING ARE PAR/PDR REFERENCE TIMEOUTS
1287   002472  013540           DH201           ;ADDRESS TESTNO ERRORPC
1288
1289                            ;ITEM 301
1290   002474  014620           DT201           ;$REG0,TESTNO,$ERRPC,0
1291   002476  015144           DF201           ;0,0,0
1292
1293                            ;ITEM 202
1294   002500  007575           EM202           ;THE FOLLOWING ARE DUAL ADDRESSING ERRORS FOR PAR'S/PDR'S
1295   002502  013567           DH202           ;ADDRESS ADDRESS
1296                                            ;LOADED  JUST READ TESTNO ERRORPC
1297
1298                            ;ITEM 302
1299   002504  014630           DT202           ;$REG0,$REG1,TESTNO,$ERRPC,0
1300   002506  015147           DF202           ;0,0,0,0
1301
1302                            ;ITEM 203
1303   002510  007666           EM203           ;THE FOLLOWING ARE COUNT PATTERN ERRORS FOR PAR'S/PDR'S
1304   002512  013647           DH203           ;ADDRESS RECEIVD  EXPECTD COUNT   TESTNO ERRORPC
1305
1306                            ;ITEM 303
1307   002514  014642           DT203           ;$REG0,$REG2,$REG4,$REG1,TESTNO,$ERRPC,0
1308   002516  015153           DF203           ;0,0,0,0,0,0
1309
1310
1311                            .SBTTL  ERROR TABLE MESSAGES AND DATA POINTERS
1312   002520  047125  054105  042520  EM1:    .ASCIZ  ?UNEXPECTED CPU TRAP OR ABORT THRU 'ERRVEC' (004)?
1313   002526  052103  042105  041440
1314   002534  052520  052040  040522
1315   002542  020120  051117  040440
1316   002550  047502  052122  052040
1317   002556  051110  020125  042447
1318   002564  051122  042526  023503
1319   002572  024040  030060  024464
```

```
1320  002600     000
1321  002601     125  042516  050130  EM3:    .ASCII  ?UNEXPECTED CACHE PARITY ERROR THRU 'CACHVEC' (114)?<CRLF>
1322  002606  041505  042524  020104
1323  002614  040503  044103  020105
1324  002622  040520  044522  054524
1325  002630  042440  051122  051117
1326  002636  052040  051110  020125
1327  002644  041447  041501  053110
1328  002652  041505  020047  030450
1329  002660  032061  100051
1330  002664  044527  046114  051040          .ASCIZ  ?WILL RETRY THIS TEST ONCE.?
1331  002672  052105  054522  052040
1332  002700  044510  020123  042524
1333  002706  052123  047440  041516
1334  002714  027105     000
1335  002717     125  042516  050130  EM4:    .ASCII  ?UNEXPECTED MAIN MEMORY PARITY ERROR THRU 'CACHVEC' (114)?<CRLF>
1336  002724  041505  042524  020104
1337  002732  040515  047111  046440
1338  002740  046505  051117  020131
1339  002746  040520  044522  054524
1340  002754  042440  051122  051117
1341  002762  052040  051110  020125
1342  002770  041447  041501  053110
1343  002776  041505  020047  030450
1344  003004  032061  100051
1345  003010  044527  046114  051040          .ASCIZ  ?WILL RETRY THIS TEST ONCE?
1346  003016  052105  054522  052040
1347  003024  044510  020123  042524
1348  003032  052123  047440  041516
1349  003040  000105
1350
1351  003042  047125  054105  042520  EM5:    .ASCIZ  ?UNEXPECTED MEMORY MANAGEMENT TRAP OR ABORT?
1352  003050  052103  042105  046440
1353  003056  046505  051117  020131
1354  003064  040515  040516  042507
1355  003072  042515  052116  052040
1356  003100  040522  020120  051117
1357  003106  040440  047502  052122
1358  003114     000
1359  003115     115  046505  051117  EM6:     .ASCIZ  ?MEMORY MANAGEMENT TRAP OR ABORT HAD INCORRECT CONDITION?
1360  003122  020131  040515  040516
1361  003130  042507  042515  052116
1362  003136  052040  040522  020120
1363  003144  051117  040440  047502
1364  003152  052122  044040  042101
1365  003160  044440  041516  051117
1366  003166  042522  052103  041440
1367  003174  047117  044504  044524
1368  003202  047117     000
1369  003205     115  046505  051117  EM7:    .ASCIZ  ?MEMORY MANAGEMENT REGISTER 0 WILL NOT CLEAR?
1370  003212  020131  040515  040516
1371  003220  042507  042515  052116
1372  003226  051040  043505  051511
1373  003234  042524  020122  020060
1374  003242  044527  046114  047040
1375  003250  052117  041440  042514
```

```
1376  003256  051101      000
1377  003261      103  047101  052047  EM10:   .ASCIZ  ?CAN'T SET 160000 IN MMR0?
1378  003266  051440  052105  030440
1379  003274  030066  030060  020060
1380  003302  047111  046440  051115
1381  003310  000060
1382  003312  047507  020124  044124  EM11:   .ASCIZ  ?GOT THE WRONG DATA BACK FROM MMR0?
1383  003320  020105  051127  047117
1384  003326  020107  040504  040524
1385  003334  041040  041501  020113
1386  003342  051106  046517  046440
1387  003350  051115  000060
1388  003354  046515  031122  042040  EM12:   .ASCIZ  ?MMR2 DID NOT TRACK PROPERLY?
1389  003362  042111  047040  052117
1390  003370  052040  040522  045503
1391  003376  050040  047522  042520
1392  003404  046122  000131
1393  003410  052523  046515  051101  EM13:   .ASCIZ  ?SUMMARY OF PAR/PDR REFERENCE TIMEOUTS?
1394  003416  020131  043117  050040
1395  003424  051101  050057  051104
1396  003432  051040  043105  051105
1397  003440  047105  042503  052040
1398  003446  046511  047505  052125
1399  003454  000123
1400  003456  047506  046114  053517  EM14:   .ASCIZ  ?FOLLOWING PAR/PDR WILL NOT CLEAR?
1401  003464  047111  020107  040520
1402  003472  027522  042120  020122
1403  003500  044527  046114  047040
1404  003506  052117  041440  042514
1405  003514  051101      000
1406  003517      123  046525  040515  EM15:   .ASCIZ  ?SUMMARY OF DUAL ADDRESSING ERRORS?
1407  003524  054522  047440  020106
1408  003532  052504  046101  040440
1409  003540  042104  042522  051523
1410  003546  047111  020107  051105
1411  003554  047522  051522      000
1412  003561      123  046525  040515  EM16:   .ASCIZ  ?SUMMARY OF COUNT PATTERN FAILURES?
1413  003566  054522  047440  020106
1414  003574  047503  047125  020124
1415  003602  040520  052124  051105
1416  003610  020116  040506  046111
1417  003616  051125  051505      000
1418  003623      105  051122  051117  EM17:   .ASCIZ  ?ERROR IN BYTE ADDRESSING OF PAR/PDR?
1419  003630  044440  020116  054502
1420  003636  042524  040440  042104
1421  003644  042522  051523  047111
1422  003652  020107  043117  050040
1423  003660  051101  050057  051104
1424  003666      000
1425  003667      117  042516  047440  EM20:   .ASCIZ  ?ONE OF THE REGISTERS TIMED OUT?
1426  003674  020106  044124  020105
1427  003702  042522  044507  052123
1428  003710  051105  020123  044524
1429  003716  042515  020104  052517
1430  003724  000124
1431  003726  052123  041501  020113  EM21:   .ASCIZ  ?STACK LIMIT REGISTER WOULD NOT CLEAR?
```

```
1432  003734  044514  044515  020124
1433  003742  042522  044507  052123
1434  003750  051105  053440  052517
1435  003756  042114  047040  052117
1436  003764  041440  042514  051101
1437  003772     000
1438  003773     105  051122  051117  EM22:   .ASCIZ  ?ERROR LOG REG. DID NOT HAVE CORRECT BIT SET?
1439  004000  046040  043517  051040
1440  004006  043505  020056  044504
1441  004014  020104  047516  020124
1442  004022  040510  042526  041440
1443  004030  051117  042522  052103
1444  004036  041040  052111  051440
1445  004044  052105     000
1446  004047     114  053517  051105  EM23:   .ASCIZ  ?LOWER BYTE OF P.S.W. NOT CORRECT?
1447  004054  041040  052131  020105
1448  004062  043117  050040  051456
1449  004070  053456  020056  047516
1450  004076  020124  047503  051122
1451  004104  041505  000124
1452  004110  046515  030522  042040  EM24:   .ASCIZ  ?MMR1 DID NOT READ ALL ZEROES?
1453  004116  042111  047040  052117
1454  004124  051040  040505  020104
1455  004132  046101  020114  042532
1456  004140  047522  051505     000
1457  004145     127  047522  043516  EM25:   .ASCIZ  ?WRONG DATA BACK FROM STACK LIMIT REGISTER?
1458  004152  042040  052101  020101
1459  004160  040502  045503  043040
1460  004166  047522  020115  052123
1461  004174  041501  020113  044514
1462  004202  044515  020124  042522
1463  004210  044507  052123  051105
1464  004216     000
1465  004217     114  040517  042504  EM26:   .ASCIZ  ?LOADED MORE THAN UPPER BYTE OF STACK LIMIT REGISTER?
1466  004224  020104  047515  042522
1467  004232  052040  040510  020116
1468  004240  050125  042520  020122
1469  004246  054502  042524  047440
1470  004254  020106  052123  041501
1471  004262  020113  044514  044515
1472  004270  020124  042522  044507
1473  004276  052123  051105     000
1474  004303     113  051105  042516  EM27:   .ASCIZ  ?KERNEL STACK POINTER NOT 1100 AFTER LOADING USP?
1475  004310  020114  052123  041501
1476  004316  020113  047520  047111
1477  004324  042524  020122  047516
1478  004332  020124  030461  030060
1479  004340  040440  052106  051105
1480  004346  046040  040517  044504
1481  004354  043516  052440  050123
1482  004362     000
1483  004363     104  040525  020114  EM30:   .ASCIZ  ?DUAL ADDRESSING BETWEEN PAR/PDR GROUPS?
1484  004370  042101  051104  051505
1485  004376  044523  043516  041040
1486  004404  052105  042527  047105
1487  004412  050040  051101  050057
```

```
1488  004420  051104  043440  047522
1489  004426  050125  000123
1490  004432  040502  020104  042522  EM31:   .ASCIZ  ?BAD RELOCATION, ON STORING DATA 18-BIT MAPPING?
1491  004440  047514  040503  044524
1492  004446  047117  020054  047117
1493  004454  051440  047524  044522
1494  004462  043516  042040  052101
1495  004470  020101  034061  041055
1496  004476  052111  046440  050101
1497  004504  044520  043516     000
1498  004511     061  026470  044502  EM32:   .ASCIZ  ?18-BIT MAPPING POSSIBLE HOLE IN MAIN MEMORY FROM?
1499  004516  020124  040515  050120
1500  004524  047111  020107  047520
1501  004532  051523  041111  042514
1502  004540  044040  046117  020105
1503  004546  047111  046440  044501
1504  004554  020116  042515  047515
1505  004562  054522  043040  047522
1506  004570  000115
1507  004572  040506  046125  054524  EM33:   .ASCIZ  ?FAULTY CARRY PROPAGATION 18-BIT MAPPING.?
1508  004600  041440  051101  054522
1509  004606  050040  047522  040520
1510  004614  040507  044524  047117
1511  004622  030440  026470  044502
1512  004630  020124  040515  050120
1513  004636  047111  027107     000
1514  004643     104  042111  023516  EM34:   .ASCIZ  ?DIDN'T GET WRAP AROUND TO ADDRESS ZERO?
1515  004650  020124  042507  020124
1516  004656  051127  050101  040440
1517  004664  047522  047125  020104
1518  004672  047524  040440  042104
1519  004700  042522  051523  055040
1520  004706  051105  000117
1521  004712  040520  042507  046040  EM35:   .ASCIZ  ?PAGE LENGTH ABORT AT WRONG VIRTUAL ADDRESS?
1522  004720  047105  052107  020110
1523  004726  041101  051117  020124
1524  004734  052101  053440  047522
1525  004742  043516  053040  051111
1526  004750  052524  046101  040440
1527  004756  042104  042522  051523
1528  004764     000
1529  004765     116  020117  040520  EM36:   .ASCIZ  ?NO PAGE LENGTH ABORT, WHEN CONDITION CORRECT?
1530  004772  042507  046040  047105
1531  005000  052107  020110  041101
1532  005006  051117  026124  053440
1533  005014  042510  020116  047503
1534  005022  042116  052111  047511
1535  005030  020116  047503  051122
1536  005036  041505  000124
1537  005042  044504  020104  047516  EM37:   .ASCIZ  ?DID NOT ABORT ON ACCESS OF NON-RESIDENT PAGE THRU KIPDR5?
1538  005050  020124  041101  051117
1539  005056  020124  047117  040440
1540  005064  041503  051505  020123
1541  005072  043117  047040  047117
1542  005100  051055  051505  042111
1543  005106  047105  020124  040520
```

```
1544  005114  042507  052040  051110
1545  005122  020125  044513  042120
1546  005130  032522     000
1547  005133     104  042111  047040  EM40:   .ASCIZ  ?DID NOT ABORT ON WRITE TO READ ONLY PAGE THRU KIPDR4?
1548  005140  052117  040440  047502
1549  005146  052122  047440  020116
1550  005154  051127  052111  020105
1551  005162  047524  051040  040505
1552  005170  020104  047117  054514
1553  005176  050040  043501  020105
1554  005204  044124  052522  045440
1555  005212  050111  051104  000064
1556  005220  047520  047111  042524  EM41:   .ASCIZ  ?POINTER VALUE 400 CAUSED STACK VIOLATION?
1557  005226  020122  040526  052514
1558  005234  020105  030064  020060
1559  005242  040503  051525  042105
1560  005250  051440  040524  045503
1561  005256  053040  047511  040514
1562  005264  044524  047117     000
1563  005271     131  046105  047514  EM42:   .ASCIZ  ?YELLOW ZONE VIOLATION EXPECTED-GOT NONE?
1564  005276  020127  047532  042516
1565  005304  053040  047511  040514
1566  005312  044524  047117  042440
1567  005320  050130  041505  042524
1568  005326  026504  047507  020124
1569  005334  047516  042516     000
1570  005341     131  046105  047514  EM43:   .ASCIZ  ?YELLOW ZONE VIOLATION EXPECTED-GOT RED?
1571  005346  020127  047532  042516
1572  005354  053040  047511  040514
1573  005362  044524  047117  042440
1574  005370  050130  041505  042524
1575  005376  026504  047507  020124
1576  005404  042522  000104
1577  005410  042522  020104  047532  EM44:   .ASCIZ  ?RED ZONE VIOLATION EXPECTED-GOT NONE?
1578  005416  042516  053040  047511
1579  005424  040514  044524  047117
1580  005432  042440  050130  041505
1581  005440  042524  026504  047507
1582  005446  020124  047516  042516
1583  005454     000
1584  005455     122  042105  055040  EM45:   .ASCIZ  ?RED ZONE VIOLATION DID NOT CAUSE ABORT?
1585  005462  047117  020105  044526
1586  005470  046117  052101  047511
1587  005476  020116  044504  020104
1588  005504  047516  020124  040503
1589  005512  051525  020105  041101
1590  005520  051117  000124
1591  005524  042522  020104  047532  EM46:   .ASCIZ  ?RED ZONE VIOLATION EXPECTED - GOT YELLOW?
1592  005532  042516  053040  047511
1593  005540  040514  044524  047117
1594  005546  042440  050130  041505
1595  005554  042524  020104  020055
1596  005562  047507  020124  042531
1597  005570  046114  053517     000
1598  005575     105  051122  051117  EM47:   .ASCII  ?ERROR DURING ERROR-ON-ERROR TRAP SEQUENCE?<CRLF>
1599  005602  042040  051125  047111
```

```
  1600  005610  020107  051105  047522
  1601  005616  026522  047117  042455
  1602  005624  051122  051117  052040
  1603  005632  040522  020120  042523
  1604  005640  052521  047105  042503
  1605  005646     200
  1606  005647     105  050130  041505          .ASCIZ  ?EXPECTED:?
  1607  005654  042524  035104     000
  1608  005661     101  020124  042514  EM50:   .ASCIZ  ?AT LEAST ONE M.M. STATUS REGISTERS WAS CLOCKED AFTER BEING LOCKED?
  1609  005666  051501  020124  047117
  1610  005674  020105  027115  027115
  1611  005702  051440  040524  052524
  1612  005710  020123  042522  044507
  1613  005716  052123  051105  020123
  1614  005724  040527  020123  046103
  1615  005732  041517  042513  020104
  1616  005740  043101  042524  020122
  1617  005746  042502  047111  020107
  1618  005754  047514  045503  042105
  1619  005762     000
  1620  005763     104  042111  047040  EM51:   .ASCIZ  ?DID NOT CHANGE MAPPING TO USER MODE?
  1621  005770  052117  041440  040510
  1622  005776  043516  020105  040515
  1623  006004  050120  047111  020107
  1624  006012  047524  052440  042523
  1625  006020  020122  047515  042504
  1626  006026     000
  1627  006027     101  047502  052122  EM52:   .ASCIZ  ?ABORT CONDITION INCORRECT EXPECTING 100153?
  1628  006034  041440  047117  044504
  1629  006042  044524  047117  044440
  1630  006050  041516  051117  042522
  1631  006056  052103  042440  050130
  1632  006064  041505  044524  043516
  1633  006072  030440  030060  032461
  1634  006100  000063
  1635  006102  046515  031122  046040  EM53:   .ASCIZ  ?MMR2 LOCKED UP THE WRONG VIRTUAL ADDRESS?
  1636  006110  041517  042513  020104
  1637  006116  050125  052040  042510
  1638  006124  053440  047522  043516
  1639  006132  053040  051111  052524
  1640  006140  046101  040440  042104
  1641  006146  042522  051523     000
  1642  006153     115  051115  020060  EM54:   .ASCIZ  ?MMR0 LOCKED UP THE WRONG PAGE NUMBER?
  1643  006160  047514  045503  042105
  1644  006166  052440  020120  044124
  1645  006174  020105  051127  047117
  1646  006202  020107  040520  042507
  1647  006210  047040  046525  042502
  1648  006216  000122
  1649  006220  026527  044502  020124  EM55:   .ASCIZ  ?W-BIT NOT SET ON WRITE TO PAGE 4?
  1650  006226  047516  020124  042523
  1651  006234  020124  047117  053440
  1652  006242  044522  042524  052040
  1653  006250  020117  040520  042507
  1654  006256  032040     000
  1655  006261     127  041055  052111  EM56:   .ASCIZ  ?W-BIT DID NOT REMAIN SET ON M.M. ABORT AT PAGE 4?
```

```
1656  006266  042040  042111  047040
1657  006274  052117  051040  046505
1658  006302  044501  020116  042523
1659  006310  020124  047117  046440
1660  006316  046456  020056  041101
1661  006324  051117  020124  052101
1662  006332  050040  043501  020105
1663  006340  000064
1664  006342  026527  044502  020124  EM57:   .ASCIZ  ?W-BIT DID NOT CLEAR ON WRITE TO KIPDR4?
1665  006350  044504  020104  047516
1666  006356  020124  046103  040505
1667  006364  020122  047117  053440
1668  006372  044522  042524  052040
1669  006400  020117  044513  042120
1670  006406  032122     000
1671  006411     127  041055  052111  EM60:   .ASCIZ  ?W-BIT WRONG ON WRITE TO PSW?
1672  006416  053440  047522  043516
1673  006424  047440  020116  051127
1674  006432  052111  020105  047524
1675  006440  050040  053523     000
1676  006445     127  041055  052111  EM61:   .ASCIZ  ?W-BIT SET AFTER WRITE TO PDR 4?
1677  006452  051440  052105  040440
1678  006460  052106  051105  053440
1679  006466  044522  042524  052040
1680  006474  020117  042120  020122
1681  006502  000064
1682  006504  052504  046101  046440  EM62:   .ASCIZ  ?DUAL MAPPING BETWEEN PAGES?
1683  006512  050101  044520  043516
1684  006520  041040  052105  042527
1685  006526  047105  050040  043501
1686  006534  051505     000
1687  006537     116  020117  040520  EM63:   .ASCIZ  ?NO PAGE HAD ITS BIT SET?
1688  006544  042507  044040  042101
1689  006552  044440  051524  041040
1690  006560  052111  051440  052105
1691  006566     000
1692  006567     104  042111  047040  EM64:   .ASCIZ  ?DID NOT PICK UP CORRECT STACK POINTER?
1693  006574  052117  050040  041511
1694  006602  020113  050125  041440
1695  006610  051117  042522  052103
1696  006616  051440  040524  045503
1697  006624  050040  044517  052116
1698  006632  051105     000
1699  006635     103  051125  042522  EM65:   .ASCIZ  ?CURRENT MODE STACK NOT PUSHED IN MFP INSTRUCTION?
1700  006642  052116  046440  042117
1701  006650  020105  052123  041501
1702  006656  020113  047516  020124
1703  006664  052520  044123  042105
1704  006672  044440  020116  043115
1705  006700  020120  047111  052123
1706  006706  052522  052103  047511
1707  006714  000116
1708  006716  051127  047117  020107  EM66:   .ASCIZ  ?WRONG DATA FETCHED BY MFP INSTRUCTION?
1709  006724  040504  040524  043040
1710  006732  052105  044103  042105
1711  006740  041040  020131  043115
```

```
1712  006746  020120  047111  052123
1713  006754  052522  052103  047511
1714  006762  000116
1715  006764  051124  042511  020104  EM67:   .ASCIZ  ?TRIED TO REFERENCE NON-RESIDENT PAGE?
1716  006772  047524  051040  043105
1717  007000  051105  047105  042503
1718  007006  047040  047117  051055
1719  007014  051505  042111  047105
1720  007022  020124  040520  042507
1721  007030     000
1722  007031     123  040524  045503  EM70:   .ASCIZ  ?STACK POINTER NOT CHANGED BY MTP INSTRUCTION?
1723  007036  050040  044517  052116
1724  007044  051105  047040  052117
1725  007052  041440  040510  043516
1726  007060  042105  041040  020131
1727  007066  052115  020120  047111
1728  007074  052123  052522  052103
1729  007102  047511  000116
1730  007106  047111  047503  051122  EM71:   .ASCIZ  ?INCORRECT STORE BY MTP INSTRUCTION?
1731  007114  041505  020124  052123
1732  007122  051117  020105  054502
1733  007130  046440  050124  044440
1734  007136  051516  051124  041525
1735  007144  044524  047117     000
1736  007151     101  047502  052122  EM72:   .ASCIZ  ?ABORTED THRU RIGHT VECTOR BUT PICKED UP WRONG PSW?
1737  007156  042105  052040  051110
1738  007164  020125  044522  044107
1739  007172  020124  042526  052103
1740  007200  051117  041040  052125
1741  007206  050040  041511  042513
1742  007214  020104  050125  053440
1743  007222  047522  043516  050040
1744  007230  053523     000
1745  007233     101  047502  052122  EM73:   .ASCIZ  ?ABORTED THRU USER SPACE, USER PSW IS XXXDDD?
1746  007240  042105  052040  051110
1747  007246  020125  051525  051105
1748  007254  051440  040520  042503
1749  007262  020054  051525  051105
1750  007270  050040  053523  044440
1751  007276  020123  054130  030130
1752  007304  030060     000
1753  007307     104  042111  047040  EM74:   .ASCIZ  ?DID NOT ABORT REFERENCE TO A MEMORY MANAGEMENT REGISTER?
1754  007314  052117  040440  047502
1755  007322  052122  051040  043105
1756  007330  051105  047105  042503
1757  007336  052040  020117  020101
1758  007344  042515  047515  054522
1759  007352  046440  047101  043501
1760  007360  046505  047105  020124
1761  007366  042522  044507  052123
1762  007374  051105     000
1763  007377     103  052520  042440  EM75:   .ASCIZ  ?CPU ERROR REG. DID NOT REPORT YELLOW ZONE?
1764  007404  051122  051117  051040
1765  007412  043505  020056  044504
1766  007420  020104  047516  020124
1767  007426  042522  047520  052122
```

```
1768  007434  054440  046105  047514
1769  007442  020127  047532  042516
1770  007450     000
1771  007451     103  052520  042440  EM76:   .ASCIZ  ?CPU ERROR REG. DID NOT REPORT RED ZONE?
1772  007456  051122  051117  051040
1773  007464  043505  020056  044504
1774  007472  020104  047516  020124
1775  007500  042522  047520  052122
1776  007506  051040  042105  055040
1777  007514  047117  000105
1778
1779  007520  044124  020105  047506  EM201:  .ASCIZ  ?THE FOLLOWING ARE PAR/PDR REFERENCE TIMEOUTS?
1780  007526  046114  053517  047111
1781  007534  020107  051101  020105
1782  007542  040520  027522  042120
1783  007550  020122  042522  042506
1784  007556  042522  041516  020105
1785  007564  044524  042515  052517
1786  007572  051524     000
1787  007575     124  042510  043040  EM202:  .ASCIZ  ?THE FOLLOWING ARE DUAL ADDRESSING ERRORS FOR PAR'S/PDR'S?
1788  007602  046117  047514  044527
1789  007610  043516  040440  042522
1790  007616  042040  040525  020114
1791  007624  042101  051104  051505
1792  007632  044523  043516  042440
1793  007640  051122  051117  020123
1794  007646  047506  020122  040520
1795  007654  023522  027523  042120
1796  007662  023522  000123
1797  007666  044124  020105  047506  EM203:  .ASCIZ  ?THE FOLLOWING ARE COUNT PATTERN ERRORS FOR PAR'S/PDR'S?
1798  007674  046114  053517  047111
1799  007702  020107  051101  020105
1800  007710  047503  047125  020124
1801  007716  040520  052124  051105
1802  007724  020116  051105  047522
1803  007732  051522  043040  051117
1804  007740  050040  051101  051447
1805  007746  050057  051104  051447
1806  007754     000
1807
1808
1809  007755     124  051505  047124  DH1:    .ASCII  ?TESTNO  PC AT ABORT?
1810  007762  020117  050040  020103
1811  007770  052101  040440  047502
1812  007776  052122
1813  010000  050103  020125  051105  DH2:    .ASCIZ  ?CPU ERR TESTNO  PC AT ABORT?
1814  010006  020122  042524  052123
1815  010014  047516  020040  041520
1816  010022  040440  020124  041101
1817  010030  051117  000124
1818  010034  042515  020115  051105  DH3:    .ASCII  ?MEM ERR CONTROL HITMISS  ?<CRLF>
1819  010042  020122  047503  052116
1820  010050  047522  020114  044510
1821  010056  046524  051511  020123
1822  010064  100040
1823  010066  042522  044507  052123          .ASCIZ  ?REGISTR REGISTR REGISTR TESTNO  PC AT ABORT?
```

```
1824  010074  020122  042522  044507
1825  010102  052123  020122  042522
1826  010110  044507  052123  020122
1827  010116  042524  052123  047516
1828  010124  020040  041520  040440
1829  010132  020124  041101  051117
1830  010140  000124
1831  010142  051105  047522  020122  DH5:    .ASCII  ?ERROR   VIRTUAL?<CRLF>
1832  010150  020040  044526  052122
1833  010156  040525  100114
1834  010162  042522  044507  052123          .ASCIZ  ?REGISTR ADDRESS TESTNO  PC AT ABORT?
1835  010170  020122  042101  051104
1836  010176  051505  020123  042524
1837  010204  052123  047516  020040
1838  010212  041520  040440  020124
1839  010220  041101  051117  000124
1840  010226  054105  042520  052103  DH6:    .ASCII  ?EXPECTD ERROR   VIRTUAL?<CRLF>
1841  010234  020104  051105  047522
1842  010242  020122  020040  044526
1843  010250  052122  040525  100114
1844  010256  047503  042116  052111          .ASCIZ  ?CONDITN REGISTR ADDRESS TESTNO  PC AT ABORT?
1845  010264  020116  042522  044507
1846  010272  052123  020122  042101
1847  010300  051104  051505  020123
1848  010306  042524  052123  047516
1849  010314  020040  041520  040440
1850  010322  020124  041101  051117
1851  010330  000124
1852  010332  046450  051115  024460  DH7:    .ASCIZ  ?(MMR0)  TESTNO  ERRORPC?
1853  010340  020040  042524  052123
1854  010346  047516  020040  051105
1855  010354  047522  050122  000103
1856  010362  047514  042101  042105  DH11:   .ASCIZ  ?LOADED  RECEIVD TESTNO  ERRORPC?
1857  010370  020040  042522  042503
1858  010376  053111  020104  042524
1859  010404  052123  047516  020040
1860  010412  051105  047522  050122
1861  010420  000103
1862  010422  054105  042520  052103  DH12:   .ASCIZ  ?EXPECTD (MMR2)  TESTNO  ERRORPC?
1863  010430  020104  046450  051115
1864  010436  024462  020040  042524
1865  010444  052123  047516  020040
1866  010452  051105  047522  050122
1867  010460  000103
1868  010462  042101  051104  051117  DH13:   .ASCIZ  ?ADDROR  ADDRAND TESTNO  ERRORPC  #ERRORS?
1869  010470  020040  042101  051104
1870  010476  047101  020104  042524
1871  010504  052123  047516  020040
1872  010512  051105  047522  050122
1873  010520  020103  021440  051105
1874  010526  047522  051522     000
1875  010533     101  042104  042522  DH14:   .ASCIZ  ?ADDRESS DATA    TESTNO  ERRORPC?
1876  010540  051523  042040  052101
1877  010546  020101  020040  052040
1878  010554  051505  047124  020117
1879  010562  042440  051122  051117
```

```
1880  010570  041520     000
1881  010573     101  042104  047522  DH15:   .ASCII  ?ADDROR  ADDRAND ADDROR  ADDRAND?<CRLF>
1882  010600  020122  040440  042104
1883  010606  040522  042116  040440
1884  010614  042104  047522  020122
1885  010622  040440  042104  040522
1886  010630  042116     200
1887  010633     114  040517  042504          .ASCIZ  ?LOADED  LOADED  ENABLED ENABLED TESTNO ERRORPC #ERRORS?
1888  010640  020104  046040  040517
1889  010646  042504  020104  042440
1890  010654  040516  046102  042105
1891  010662  042440  040516  046102
1892  010670  042105  052040  051505
1893  010676  047124  020117  051105
1894  010704  047522  050122  020103
1895  010712  042443  051122  051117
1896  010720  000123
1897  010722  042101  051104  051117  DH16:   .ASCIZ  ?ADDROR  ADDRAND PATRNOR PATRNAD DATAOR  DATAND  TESTNO ERRORPC #ERRORS?
1898  010730  020040  042101  051104
1899  010736  047101  020104  040520
1900  010744  051124  047516  020122
1901  010752  040520  051124  040516
1902  010760  020104  040504  040524
1903  010766  051117  020040  040504
1904  010774  040524  042116  020040
1905  011002  042524  052123  047516
1906  011010  042440  051122  051117
1907  011016  041520  021440  051105
1908  011024  047522  051522     000
1909  011031     101  042104  042522  DH17:   .ASCIZ  ?ADDRESS EXPECTD RECEIVD TESTNO  ERRORPC?
1910  011036  051523  042440  050130
1911  011044  041505  042124  051040
1912  011052  041505  044505  042126
1913  011060  052040  051505  047124
1914  011066  020117  042440  051122
1915  011074  051117  041520     000
1916  011101     122  043505  042101  DH20:   .ASCIZ  ?REGADDR TESTNO  ERRORPC?
1917  011106  051104  052040  051505
1918  011114  047124  020117  042440
1919  011122  051122  051117  041520
1920  011130     000
1921  011131     122  043505  042101  DH21:   .ASCIZ  ?REGADDR DATA    TESTNO  ERRORPC?
1922  011136  051104  042040  052101
1923  011144  020101  020040  052040
1924  011152  051505  047124  020117
1925  011160  042440  051122  051117
1926  011166  041520     000
1927  011171     122  043505  042101  DH22:   .ASCIZ  ?REGADR  EXPECTD RECEIVD TESTNO ERRORPC?
1928  011176  020122  042440  050130
1929  011204  041505  042124  051040
1930  011212  041505  044505  042126
1931  011220  052040  051505  047124
1932  011226  020117  051105  047522
1933  011234  050122  000103
1934  011240  042522  040507  042104  DH23:   .ASCIZ  ?REGADDR DATAREC DATAEXP TESTNO  ERRORPC?
1935  011246  020122  040504  040524
```

```
1936  011254  042522  020103  040504
1937  011262  040524  054105  020120
1938  011270  042524  052123  047516
1939  011276  020040  051105  047522
1940  011304  050122  000103
1941  011310  051513  020120  020040  DH27:   .ASCIZ  ?KSP     TESTNO  ERRORPC?
1942  011316  020040  042524  052123
1943  011324  047516  020040  051105
1944  011332  047522  050122  000103
1945  011340  047111  042504  020130  DH30:   .ASCII  ?INDEX   INDEX   PAR/PDR?<CRLF>
1946  011346  020040  047111  042504
1947  011354  020130  020040  040520
1948  011362  027522  042120  100122
1949  011370  054105  042520  052103          .ASCIZ  ?EXPECTD RECEIVD ADRREAD TESTNO  ERRORPC?
1950  011376  020104  042522  042503
1951  011404  053111  020104  042101
1952  011412  051122  040505  020104
1953  011420  042524  052123  047516
1954  011426  020040  051105  047522
1955  011434  050122  000103
1956  011440  044526  052122  040525  DH31:   .ASCII  ?VIRTUAL         PHYSCAL DATA    DATA?<CRLF>
1957  011446  020114  020040  020040
1958  011454  020040  020040  044120
1959  011462  051531  040503  020114
1960  011470  040504  040524  020040
1961  011476  020040  040504  040524
1962  011504     2J0
1963  011505     101  042104  042522          .ASCIZ  ?ADDRESS KIPAR4  ADDRESS EXPECTD RECEIVD TESTNO ERRORPC?
1964  011512  051523  045440  050111
1965  011520  051101  020064  040440
1966  011526  042104  042522  051523
1967  011534  042440  050130  041505
1968  011542  042124  051040  041505
1969  011550  044505  042126  052040
1970  011556  051505  047124  020117
1971  011564  051105  047522  050122
1972  011572  000103
1973  011574  052123  052122  042101  DH32:   .ASCIZ  ?STRTADR FNSHADR TESTNO  ERRORPC?
1974  011602  020122  047106  044123
1975  011610  042101  020122  042524
1976  011616  052123  047516  020040
1977  011624  051105  047522  050122
1978  011632  000103
1979  011634  040520  052124  051105  DH33:   .ASCII  ?PATTERN DATA    ADDRESS?<CRLF>
1980  011642  020116  040504  040524
1981  011650  020040  020040  042101
1982  011656  051104  051505  100123
1983  011664  047514  042101  042105          .ASCIZ  ?LOADED  FETCHED INTENDD KIPAR4  KIPAR5  TESTNO  ERRORPC?
1984  011672  020040  042506  041524
1985  011700  042510  020104  047111
1986  011706  042524  042116  020104
1987  011714  044513  040520  032122
1988  011722  020040  044513  040520
1989  011730  032522  020040  042524
1990  011736  052123  047516  020040
1991  011744  051105  047522  050122
```

```
 1992  011752  000103
 1993  011754  040504  040524  020040  DH34:   .ASCIZ  ?DATA    TESTNO  ERRORPC?
 1994  011762  020040  042524  052123
 1995  011770  047516  020040  051105
 1996  011776  047522  050122  000103
 1997  012004  043520  042514  043116  DH35:   .ASCIZ  ?PGLENFD VABLKNO TESTNO  ERRORPC?
 1998  012012  020104  040526  046102
 1999  012020  047113  020117  042524
 2000  012026  052123  047516  020040
 2001  012034  051105  047522  050122
 2002  012042  000103
 2003  012044  044513  042120  032522  DH37:   .ASCIZ  ?KIPDR5  KIPAR5  VIRTADR TESTNO  ERRORPC?
 2004  012052  020040  044513  040520
 2005  012060  032522  020040  044526
 2006  012066  052122  042101  020122
 2007  012074  042524  052123  047516
 2008  012102  020040  051105  047522
 2009  012110  050122  000103
 2010  012114  044513  042120  032122  DH40:   .ASCIZ  ?KIPDR4  KIPAR4  VIRTADR TESTNO  ERRORPC?
 2011  012122  020040  044513  040520
 2012  012130  032122  020040  044526
 2013  012136  052122  042101  020122
 2014  012144  042524  052123  047516
 2015  012152  020040  051105  047522
 2016  012160  050122  000103
 2017  012164  052123  050113  051124  DH41:   .ASCIZ  ?STKPTR  STKLMT  TESTNO  ERRORPC?
 2018  012172  020040  052123  046113
 2019  012200  052115  020040  042524
 2020  012206  052123  047516  020040
 2021  012214  051105  047522  050122
 2022  012222  000103
 2023  012224  051520  004527  051411  DH47:   .ASCII  ?PSW            STK PTR (MMR0)  (MMR2)  TESTNO  ERRORPC?<CRLF>
 2024  012232  045524  050040  051124
 2025  012240  024040  046515  030122
 2026  012246  020051  024040  046515
 2027  012254  031122  020051  052040
 2028  012262  051505  047124  020117
 2029  012270  042440  051122  051117
 2030  012276  041520     200
 2031  012301     061  030064  030460          .ASCII  ?140017 -11/6X- 000000  020151?<CRLF>
 2032  012306  020067  030455  027461
 2033  012314  054066  004455  030060
 2034  012322  030060  030060  020040
 2035  012330  031060  030460  030465
 2036  012336     200
 2037  012337     061  030064  030060          .ASCII  ?140000 -11/40- 4&1074?<CRLF>
 2038  012344  020060  030455  027461
 2039  012352  030064  020055  023064
 2040  012360  030061  032067     200
 2041  012365     122  041505  044505          .ASCII  ?RECEIVED:?<CRLF>
 2042  012372  042526  035104     200
 2043  012377     120  053523  020040          .ASCIZ  ?PSW     PC      STK PTR (MMR0)  (MMR2)  TESTNO  ERRORPC?
 2044  012404  020040  050040  020103
 2045  012412  020040  020040  051440
 2046  012420  045524  050040  051124
 2047  012426  024040  046515  030122
```

```
2048  012434  020051  024040  046515
2049  012442  031122  020051  052040
2050  012450  051505  047124  020117
2051  012456  042440  051122  051117
2052  012464  041520     000
2053  012467     117  044522  044507  DH50:   .ASCII  ?ORIGINAL DATA          NEW DATA?<CRLF>
2054  012474  040516  020114  040504
2055  012502  040524  004411  047040
2056  012510  053505  042040  052101
2057  012516  100101
2058  012520  046450  051115  024460          .ASCIZ  ?(MMR0)  (MMR2)  (MMR0)  (MMR2)  TESTNO  ERRORPC?
2059  012526  020040  046450  051115
2060  012534  024462  020040  046450
2061  012542  051115  024460  020040
2062  012550  046450  051115  024462
2063  012556  020040  042524  052123
2064  012564  047516  020040  051105
2065  012572  047522  050122  000103
2066  012600  042522  042503  053111  DH52:   .ASCIZ  ?RECEIVD (MMR2)  TESTNO  ERRORPC?
2067  012606  020104  046450  051115
2068  012614  024462  020040  042524
2069  012622  052123  047516  020040
2070  012630  051105  047522  050122
2071  012636  000103
2072  012640  044526  052122  042101  DH53:   .ASCIZ  ?VIRTADR (MMR2)  TESTNO  ERRORPC?
2073  012646  020122  046450  051115
2074  012654  024462  020040  042524
2075  012662  052123  047516  020040
2076  012670  051105  047522  050122
2077  012676  000103
2078  012700  054105  042520  052103  DH54:   .ASCIZ  ?EXPECTD (MMR0)  TESTNO  ERRORPC?
2079  012706  020104  046450  051115
2080  012714  024460  020040  042524
2081  012722  052123  047516  020040
2082  012730  051105  047522  050122
2083  012736  000103
2084  012740  044513  042120  032122  DH55:   .ASCIZ  ?KIPDR4  TESTNO  ERRORPC?
2085  012746  020040  042524  052123
2086  012754  047516  020040  051105
2087  012762  047522  050122  000103
2088  012770  044513  042120  033522  DH60:   .ASCIZ  ?KIPDR7  TESTNO  ERRORPC?
2089  012776  020040  042524  052123
2090  013004  047516  020040  051105
2091  013012  047522  050122  000103
2092  013020  042107  040520  042507  DH62:   .ASCIZ  ?GDPAGE  BDPAGE  TESTNO  ERRORPC?
2093  013026  020040  042102  040520
2094  013034  042507  020040  042524
2095  013042  052123  047516  020040
2096  013050  051105  047522  050122
2097  013056  000103
2098  013060  051524  050124  043501  DH63:   .ASCIZ  ?TSTPAGE CONTENT TESTNO  ERRORPC?
2099  013066  020105  047503  052116
2100  013074  047105  020124  042524
2101  013102  052123  047516  020040
2102  013110  051105  047522  050122
2103  013116  000103
```

ERROR TABLE MESSAGES AND DATA POINTERS

```
2104  013120  054105  042520  052103  DH64:   .ASCIZ  ?EXPECTD RECEIVD TESTNO  ERRORPC?
2105  013126  020104  042522  042503
2106  013134  053111  020104  042524
2107  013142  052123  047516  020040
2108  013150  051105  047522  050122
2109  013156  000103
2110  013160  054105  042520  052103  DH66:   .ASCIZ  ?EXPECTD RECEIVD TESTNO  ERRORPC?
2111  013166  020104  042522  042503
2112  013174  053111  020104  042524
2113  013202  052123  047516  020040
2114  013210  051105  047522  050122
2115  013216  000103
2116  013220  046450  051115  024460  DH67:   .ASCIZ  ?(MMR0)  (MMR2)  TESTNO  ERRORPC?
2117  013226  020040  046450  051115
2118  013234  024462  020040  042524
2119  013242  052123  047516  020040
2120  013250  051105  047522  050122
2121  013256  000103
2122  013260  052123  050113  051124  DH70:   .ASCIZ  ?STKPTR  TESTNO  ERRORPC?
2123  013266  020040  042524  052123
2124  013274  047516  020040  051105
2125  013302  047522  050122  000103
2126  013310  042107  040504  040524  DH71:   .ASCIZ  ?GDDATA  STORED  TESTNO  ERRORPC?
2127  013316  020040  052123  051117
2128  013324  042105  020040  042524
2129  013332  052123  047516  020040
2130  013340  051105  047522  050122
2131  013346  000103
2132  013350  050050  053523  020051  DH72:   .ASCIZ  ?(PSW)   TESTNO  ERRORPC EXPECTING XXX340?
2133  013356  020040  042524  052123
2134  013364  047516  020040  051105
2135  013372  047522  050122  020103
2136  013400  054105  042520  052103
2137  013406  047111  020107  054130
2138  013414  031530  030064     000
2139  013421     050  051520  024527  DH73:   .ASCIZ  ?(PSW)   TESTNO  ERRORPC?
2140  013426  020040  052040  051505
2141  013434  047124  020117  042440
2142  013442  051122  051117  041520
2143  013450     000
2144  013451     124  051505  047124  DH74:   .ASCIZ  ?TESTNO ERRORPC?
2145  013456  020117  051105  047522
2146  013464  050122  000103
2147  013470  052123  050113  051124  DH75:   .ASCIZ  ?STKPTR  STKLMT  CPUERR  TESTNO  ERRORPC?
2148  013476  020040  052123  046113
2149  013504  052115  020040  050103
2150  013512  042525  051122  020040
2151  013520  042524  052123  047516
2152  013526  020040  051105  047522
2153  013534  050122  000103
2154
2155
2156  013540  042101  051104  051505  DH201:  .ASCIZ  ?ADDRESS TESTNO ERRORPC?
2157  013546  020123  042524  052123
2158  013554  047516  042440  051122
2159  013562  051117  041520     000
```

```
2160  013567  000101  042104  042522  DH202:  .ASCII  ?ADDRESS ADDRESS?<CRLF>
2161  013574  051523  040440  042104
2162  013602  042522  051523  000200
2163  013607  000114  040517  042504          .ASCIZ  ?LOADED  JUSTREAD TESTNO ERRORPC?
2164  013614  020104  045040  051525
2165  013622  051124  040505  020104
2166  013630  042524  052123  047516
2167  013636  042440  051122  051117
2168  013644  041520  000000
2169  013647  000101  042104  042522  DH203:  .ASCIZ  ?ADDRESS RECEIVD EXPECTD COUNT   TESTNO ERRORPC?
2170  013654  051523  051040  041505
2171  013662  044505  042126  042440
2172  013670  050130  041505  042124
2173  013676  041440  052517  052116
2174  013704  020040  052040  051505
2175  013712  047124  020117  051105
2176  013720  047522  050122  000103
2177                                          .EVEN
2178
2179
2180  013726  001414  001412  000000  DT1:    .WORD   TESTNO,BADPC,0
2181  013734  001410  001414  001412  DT2:    .WORD   PCPUER,TESTNO,BADPC,0
2182  013742  000000
2183  013744  001376  001400  001402  DT3:    .WORD   PPARER,PCONTR,PHITMI,TESTNO,BADPC,0
2184  013752  001414  001412  000000
2185  013760  001404  001406  001414  DT5:    .WORD   PMMR0,PMMR2,TESTNO,BADPC,0
2186  013766  001412  000000
2187  013772  001366  001404  001406  DT6:    .WORD   MMEXP,PMMR0,PMMR2,TESTNO,BADPC,0
2188  014000  001414  001412  000000
2189  014006  001164  001414  001116  DT7:    .WORD   $REG1,TESTNO,$ERRPC,0
2190  014014  000000
2191  014016  001166  001164  001414  DT11:   .WORD   $REG2,$REG1,TESTNO,$ERRPC,0
2192  014024  001116  000000
2193  014030  001170  001166  001414  DT12:   .WORD   $REG3,$REG2,TESTNO,$ERRPC,0
2194  014036  001116  000000
2195  014042  001426  001430  001414  DT13:   .WORD   ADDROR,ADRAND,TESTNO,$ERRPC,$TMP5,0
2196  014050  001116  001210  000000
2197  014056  001162  001166  001414  DT14:   .WORD   $REG0,$REG2,TESTNO,$ERRPC,0
2198  014064  001116  000000
2199  014070  001426  001430  001416  DT15:   .WORD   ADDROR,ADRAND,DATAOR,DATAND,TESTNO,$ERRPC,$TMP5,0
2200  014076  001420  001414  001116
2201  014104  001210  000000
2202  014110  001426  001430  001422  DT16:   .WORD   ADDROR,ADRAND,PATTOR,PATAND,DATAOR,DATAND,TESTNO,$ERRPC,$TMP5,0
2203  014116  001424  001416  001420
2204  014124  001414  001116  001210
2205  014132  000000
2206  014134  001162  001166  001164  DT17:   .WORD   $REG0,$REG2,$REG1,TESTNO,$ERRPC,0
2207  014142  001414  001116  000000
2208  014150  001162  001414  001116  DT20:   .WORD   $REG0,TESTNO,$ERRPC,0
2209  014156  000000
2210  014160  001162  001164  001414  DT21:   .WORD   $REG0,$REG1,TESTNO,$ERRPC,0
2211  014166  001116  000000
2212  014172  001206  001210  001162  DT22:   .WORD   $TMP4,$TMP5,$REG0,TESTNO,$ERRPC,0
2213  014200  001414  001116  000000
2214  014206  001162  001164  001166  DT23:   .WORD   $REG0,$REG1,$REG2,TESTNO,$ERRPC,0
2215  014214  001414  001116  000000
```

```
2216  014222  001162  001414  001116  DT27:   .WORD   $REG0,TESTNO,$ERRPC,0
2217  014230  000000
2218  014232  001162  001164  001166  DT30:   .WORD   $REG0,$REG1,$REG2,TESTNO,$ERRPC,0
2219  014240  001414  001116  000000
2220  014246  001162  001200  001202  DT31:   .WORD   $REG0,$TMP1,$TMP2,$REG1,$REG2,TESTNO,$ERRPC,0
2221  014254  001164  001166  001414
2222  014262  001116  000000
2223  014266  001200  001202  001414  DT32:   .WORD   $TMP1,$TMP2,TESTNO,$ERRPC,0
2224  014274  001116  000000
2225  014300  001166  001170  001162  DT33:   .WORD   $REG2,$REG3,$REG0,$TMP3,$TMP4,TESTNO,$ERRPC,0
2226  014306  001204  001206  001414
2227  014314  001116  000000
2228  014320  001164  001414  001116  DT34:   .WORD   $REG1,TESTNO,$ERRPC,0
2229  014326  000000
2230  014330  001164  001170  001414  DT35:   .WORD   $REG1,$REG3,TESTNO,$ERRPC,0
2231  014336  001116  000000
2232  014342  001176  001200  001202  DT37:   .WORD   $TMP0,$TMP1,$TMP2,TESTNO,$ERRPC,0
2233  014350  001414  001116  000000
2234  014356  001166  001170  001414  DT41:   .WORD   $REG2,$REG3,TESTNO,$ERRPC,0
2235  014364  001116  000000
2236  014370  001166  001170  001164  DT47:   .WORD   $REG2,$REG3,$REG1,PMMR0,PMMR2,TESTNO,$ERRPC,0
2237  014376  001404  001406  001414
2238  014404  001116  000000
2239  014410  001404  001406  001176  DT50:   .WORD   PMMR0,PMMR2,$TMP0,$TMP2,TESTNO,$ERRPC,0
2240  014416  001202  001414  001116
2241  014424  000000
2242  014426  001404  001406  001414  DT52:   .WORD   PMMR0,PMMR2,TESTNO,$ERRPC,0
2243  014434  001116  000000
2244  014440  001164  001406  001414  DT53:   .WORD   $REG1,PMMR2,TESTNO,$ERRPC,0
2245  014446  001116  000000
2246  014452  001166  001404  001414  DT54:   .WORD   $REG2,PMMR0,TESTNO,$ERRPC,0
2247  014460  001116  000000
2248  014464  001176  001414  001116  DT55:   .WORD   $TMP0,TESTNO,$ERRPC,0
2249  014472  000000
2250  014474  001170  001164  001414  DT62:   .WORD   $REG3,$REG1,TESTNO,$ERRPC,0
2251  014502  001116  000000
2252  014506  001170  001162  001414  DT63:   .WORD   $REG3,$REG0,TESTNO,$ERRPC,0
2253  014514  001116  000000
2254  014520  001166  001164  001414  DT64:   .WORD   $REG2,$REG1,TESTNO,$ERRORPC,0
2255  014526  043300  000000
2256  014532  001162  001164  001414  DT66:   .WORD   $REG0,$REG1,TESTNO,$ERRPC,0
2257  014540  001116  000000
2258  014544  001404  001406  001414  DT67:   .WORD   PMMR0,PMMR2,TESTNO,$ERRPC,0
2259  014552  001116  000000
2260  014556  001164  001414  001116  DT70:   .WORD   $REG1,TESTNO,$ERRPC,0
2261  014564  000000
2262  014566  001162  001414  001116  DT72:   .WORD   $REG0,TESTNO,$ERRPC,0
2263  014574  000000
2264  014576  001414  001116  000000  DT74:   .WORD   TESTNO,$ERRPC,0
2265  014604  001166  001170  001410  DT75:   .WORD   $REG2,$REG3,PCPUER,TESTNO,$ERRPC,0
2266  014612  001414  001116  000000
2267
2268  014620  001162  001414  001116  DT201:  .WORD   $REG0,TESTNO,$ERRPC,0
2269  014626  000000
2270  014630  001162  001164  001414  DT202:  .WORD   $REG0,$REG1,TESTNO,$ERRPC,0
2271  014636  001116  000000
```

```
2272  014642  001162  001166  001172  DT203:  .WORD   $REG0,$REG2,$REG4,$REG1,TESTNO,$ERRPC,0
2273  014650  001164  001414  001116
2274  014656  000000
2275
2276
2277  014660      000     000         DF1:    .BYTE   0,0
2278  014662      000     000     000 DF2:    .BYTE   0,0,0
2279  014665      000     000     000 DF3:    .BYTE   0,0,0,0,0
2280  014670      000     000
2281  014672      000     000     000 DF5:    .BYTE   0,0,0,0
2282  014675      000
2283  014676      000     000     000 DF6:    .BYTE   0,0,0,0,0
2284  014701      000     000
2285  014703      000     000     000 DF7:    .BYTE   0,0,0
2286  014706      000     000     000 DF11:   .BYTE   0,0,0,0
2287  014711      000
2288  014712      000     000     000 DF12:   .BYTE   0,0,0,0
2289  014715      000
2290  014716      000     000     000 DF13:   .BYTE   0,0,0,0,1
2291  014721      000     001
2292  014723      000     000     000 DF14:   .BYTE   0,0,0,0
2293  014726      000
2294  014727      000     000     000 DF15:   .BYTE   0,0,0,0,0,0,1
2295  014732      000     000     000
2296  014735      001
2297  014736      000     000     000 DF16:   .BYTE   0,0,0,0,0,0,0,0,1
2298  014741      000     000     000
2299  014744      000     000     001
2300  014747      000     000     000 DF17:   .BYTE   0,0,0,0,0
2301  014752      000     000
2302  014754      000     000     000 DF20:   .BYTE   0,0,0
2303  014757      000     000     000 DF21:   .BYTE   0,0,0,0
2304  014762      000
2305  014763      000     000     000 DF22:   .BYTE   0,0,0,0,0
2306  014766      000     000
2307  014770      000     000     000 DF23:   .BYTE   0,0,0,0,0
2308  014773      000     000
2309  014775      000     000     000 DF27:   .BYTE   0,0,0
2310  015000      000     000     000 DF30:   .BYTE   0,0,0,0,0
2311  015003      000     000
2312  015005      000     000     000 DF31:   .BYTE   0,0,0,0,0,0,0
2313  015010      000     000     000
2314  015013      000
2315  015014      002     002     000 DF32:   .BYTE   2,2,0,0
2316  015017      000
2317  015020      000     000     000 DF33:   .BYTE   0,0,0,0,0,0,0
2318  015023      000     000     000
2319  015026      000
2320  015027      000     000     000 DF34:   .BYTE   0,0,0
2321  015032      000     000     000 DF35:   .BYTE   0,0,0,0
2322  015035      000
2323  015036      000     000     000 DF37:   .BYTE   0,0,0,0,0
2324  015041      000     000
2325  015043      000     000     000 DF41:   .BYTE   0,0,0,0
2326  015046      000
2327  015047      000     000     000 DF47:   .BYTE   0,0,0,0,0,0,0
```

```
2328  015052      000      000      000
2329  015055      000
2330  015056      000      000      000  DF50:   .BYTE   0,0,0,0,0,0
2331  015061      000      000      000
2332  015064      000      000      000  DF52:   .BYTE   0,0,0,0
2333  015067      000
2334  015070      000      000      000  DF53:   .BYTE   0,0,0,0
2335  015073      000
2336  015074      000      000      000  DF54:   .BYTE   0,0,0,0
2337  015077      000
2338  015100      000      000      000  DF55:   .BYTE   0,0,0
2339  015103      000      000      000  DF62:   .BYTE   0,0,0,0
2340  015106      000
2341  015107      000      000      000  DF63:   .BYTE   0,0,0,0
2342  015112      000
2343  015113      000      000      000  DF64:   .BYTE   0,0,0,0
2344  015116      000
2345  015117      000      000      000  DF66:   .BYTE   0,0,0,0
2346  015122      000
2347  015123      000      000      000  DF67:   .BYTE   0,0,0,0
2348  015126      000
2349  015127      000      000      000  DF70:   .BYTE   0,0,0
2350  015132      000      000      000  DF72:   .BYTE   0,0,0
2351  015135      000      000           DF74:   .BYTE   0,0
2352  015137      000      000      000  DF75:   .BYTE   0,0,0,0,0
2353  015142      000      000
2354
2355
2356
2357  015144      000      000      000  DF201:  .BYTE   0,0,0
2358  015147      000      000      000  DF202:  .BYTE   0,0,0,0
2359  015152      000
2360  015153      000      000      000  DF203:  .BYTE   0,0,0,0,0,0
2361  015156      000      000      000
2362              015162                        .EVEN
2363
2364
2365
2366
2367
```

```
                                        ERROR TABLE MESSAGES AND DATA POINTERS


                                .SBTTL  ********************* TRAP HANDLING ROUTINES *********************

                                .SBTTL  CPU TRAP HANDLER ROUTINE
                                ;;**************************************************************************
                                ;*
                                ;*      THIS SUBROUTINE WILL HANDLE ALL CPU TRAPS AND ABORTS, THRU
                                ;*      "ERRVEC" (000004).  IF THIS SUBROUTINE IS ENTERED BY A SECOND
                                ;*      TRAP BEFORE THE FIRST HAS BEEN PROCESSED A HALT IS EXECUTED.
                                ;*      THE CPU ERROR REGISTER IS READ IF EXECUTING ON AN 11/6X.
                                ;*      FOR EVERY CASE AN ERROR MESSAGE IS TYPED REPORTING AN
                                ;*      UNEXPECTED TRAP OR ABORT TO LOC 4.  'PCPUER' CAN
                                ;*      BE USED AS A FLAG TO INDICATE THAT A TRAP HAS OCCURRED SINCE IT
                                ;*      IS LOADED WITH THE ERROR REGISTER IF A TRAP VECTORS HERE
                                ;*
                                ;;**************************************************************************
015162  005227          CPUER:  INC     (PC)+                   ;MAKE FLAG ZERO IF FIRST TIME
015164  177777          CPFLAG: .WORD   -1                      ;NEGATIVE ONE FOR A FLAG
015166  001401                  BEQ     10$                     ;BRANCH IF FIRST TIME IN
015170  000000                  HALT                            ;I HAVE ENTERED THIS ROUTINE BEFORE
                                                                ;I FINISHED REPORTING THE FIRST ERROR
                                                                ;THE SECOND ENTRY ADDRESS IS ON THE
                                                                ;STACK AND THE FIRST ERROR CONDITION
                                                                ;IS PROBABLY STILL LOCKED UP
015172  012637  001436    10$:  MOV     (KSP)+,OLDPC            ;SAVE RETURN ADDRESS IN CASE OF LOOP
015176  012637  001440          MOV     (KSP)+,OLDPS            ;SAVE OLD PSW IN CASE OF LOOP
015202  013737  001436  001412  MOV     OLDPC,BADPC             ;SAVE PC+2 AT TIME OF ABORT
015210  105737  001450          TSTB    FORTY                   ;EXECUTING ON AN 11/40?
015214  001402                  BEQ     2$                      ;BRANCH IF NO
015216  104001                  ERROR   1                       ;UNEXPECTED CPU TRAP OR ABORT
015220  000404                  BR      1$                      ;BRANCH TO EXIT
015222  017737  164126  001410  2$:     MOV     @CPUERR,PCPUER  ;SAVE CPU ERROR REGISTER ON AN 11/6X
015230  104002                  ERROR   2                       ;UNEXPECTED CPU TRAP OR ABORT
015232  012737  177777  015164  1$:     MOV     #-1,CPFLAG      ;MAKE FLAG NEGATIVE ONE FOR NEXT TIME
015240  013746  001440          MOV     OLDPS,-(KSP)            ;PUSH OLD PSW BACK ON STACK
015244  013746  001436          MOV     OLDPC,-(KSP)            ;PUSH RETURN ADDRESS BACK ON STACK
015250  000006                  RTT                             ;RETURN FROM INTERRUPT OR ABORT


                                .SBTTL  MEMORY/CACHE PARITY TRAPS AND ABORTS HANDLER ROUTINE
                                ;;**************************************************************************
                                ;*
                                ;*      THIS SUBROUTINE WILL HANDLE ALL UNEXPECTED PARITY ERRORS.  IF
                                ;*      THE PARITY ERROR IS AN ABORT THE TEST THAT WAS RUNNING WILL
                                ;*      BE RESTARTED ONCE AFTER THE ABORT CONDITION IS REPORTED.  ON THE
                                ;*      SECOND ABORT IN A SINGLE TEST THE NEXT TEST IS ATTEMPTED
                                ;*      AFTER THE ABORT IS REPORTED.
                                ;*
                                ;;**************************************************************************
015252  005227          MEMER:  INC     (PC)+                   ;MAKE FLAG ZERO IF FIRST TIME
015254  177777          PAFLAG: .WORD   -1                      ;NEGATIVE ONE FOR A FLAG
015256  001401                  BEQ     10$                     ;BRANCH IF FIRST TIME IN
015260  000000                  HALT                            ;I HAVE ENTERED THIS ROUTINE BEFORE
                                                                ;I FINISHED REPORTING THE FIRST ERROR
```

```
2424                                                    ;THE SECOND ENTRY ADDRESS IS ON THE
2425                                                    ;STACK AND THE FIRST ERROR CONDITION
2426                                                    ;IS PROBABLY STILL LOCKED UP .
2427  015262  012637  001436          10$:    MOV     (KSP)+,OLDPC    ;SAVE RETURN ADDRESS IN CASE OF LOOP
2428  015266  012637  001440                  MOV     (KSP)+,OLDPS    ;SAVE OLD PSW IN CASE OF LOOP
2429  015272  105737  001450                  TSTB    FORTY           ;EXECUTING ON AN 11/40?
2430  015276  001026                          BNE     6$              ;BRANCH IF YES
2431  015300  017737  164054  001376          MOV     @#MEMERR,PPARER ;SAVE MEMORY ERROR REGISTER
2432  015306  017737  164040  001400          MOV     @#CONTRL,PCONTR ;SAVE CONTROL REGISTER FOR TYPE OUT
2433  015314  017737  164036  001402          MOV     @#HITMIS,PHITMI ;SAVE HIT/MISS REGISTER
2434  015322  013737  001436  001412          MOV     OLDPC,BADPC     ;SAVE PC+2 AT TIME OF ABORT
2435  015330  032777  000014  164014          BIT     #14,@#CONTRL    ;SEE IF CACHE IS ON OR OFF
2436  015336  001010                          BNE     1$              ;IF OFF, MUST BE MEMORY REFERENCE
2437  015340  032777  000340  164030          BIT     #340,@#PPARER   ;WAS THIS A CACHE PARITY ERROR
2438  015346  001004                          BNE     1$              ;BRANCH IF IT WAS A MAIN MEMORY ERROR
2439  015350  104003                          ERROR   3               ;UNEXPECTED CACHE PARITY ERROR
2440  015352  000403                          BR      2$              ;BRANCH TO EXIT POINT
2441  015354  104077                  6$:     ERROR   77              ;UNEXPECTED MAIN MEM. PARITY ERR.  11/40
2442  015356  000401                          BR      2$              ;BRANCH TO EXIT POINT
2443  015360  104004                  1$:     ERROR   4               ;UNEXPECTED MAIN MEMORY PARITY ERROR
2444  015362  005737  001444          2$:     TST     RETRY           ;ARE YOU RETRYING THIS TEST?
2445  015366  001006                          BNE     3$              ;BRANCH IF THIS IS SECOND TRY
2446  015370  005237  001444                  INC     RETRY           ;SET RETRY FLAG
2447  015374  013737  001106  001436          MOV     $LPADR,OLDPC    ;RETURN TO START OF THIS TEST
2448  015402  000403                          BR      4$              ;BRANCH TO EXIT
2449  015404  013737  001446  001436  3$:     MOV     NXTTST,OLDPC    ;RETURN TO START OF NEXT TEST
2450                                                    ;SINCE THIS IS THE SECOND ABORT
2451  015412  012737  177777  015254  4$:     MOV     #-1,PAFLAG      ;RESTORE A NEGATIVE ONE FOR NEXT TIME
2452  015420  013746  001440                  MOV     OLDPS,-(KSP)    ;PUSH OLD PSW BACK ON STACK
2453  015424  013746  001436                  MOV     OLDPC,-(KSP)    ;PUSH RETURN ADDRESS BACK ON STACK
2454  015430  000006                  5$:     RTT                     ;RETURN FROM INTERRUPT.
2455
2456
2457
2458                                    .SBTTL  MEMORY MANAGEMENT TRAPS AND ABORTS HANDLER ROUTINE
2459                                    ;;**************************************************************************
2460                                    ;*
2461                                    ;*      THIS SUBROUTINE WILL HANDLE MOST OF THE MEMORY MANAGEMENT TRAPS
2462                                    ;*      AND ABORTS THAT ARE GENERATED DURING THIS PROGRAM.  ANY M.M.
2463                                    ;*      ABORTS OR TRAPS THAT OCCUR WHILE 'MMEXP' IS ZERO ARE UNEXPECTED
2464                                    ;*      AND WILL BE REPORTED AS SUCH.  THIS MAY OCCUR BECAUSE SOME LOGIC
2465                                    ;*      THAT IS TO INHIBIT A TRAP HAS FAILED.
2466                                    ;*      THERE ARE ALSO TIMES WHEN I AM EXPECTING A CERTAIN CONDITION TO
2467                                    ;*      OCCUR, AND WHEN THIS CONDITION IN 'MMEXP' DOES NOT MATCH THE
2468                                    ;*      CONTENTS OF 'PMMR0' (SAME AS 'MMR0') AN ERROR IS REPORTED.
2469                                    ;*
2470                                    ;;**************************************************************************
2471  015432  005227                  MMTRAP: INC     (PC)+           ;MAKE FLAG ZERO IF FIRST TIME
2472  015434  177777                  MMFLAG: .WORD   -1              ;FLAG SHOULD BE NEG ONE
2473  015436  001401                          BEQ     10$             ;BRANCH IF FIRST TIME INTO ROUTINE
2474  015440  000000                          HALT                    ;I HAVE ENTERED THIS ROUTINE BEFORE
2475                                                    ;I FINISHED REPORTING THE FIRST ERROR
2476                                                    ;THE SECOND ENTRY ADDRESS IS ON THE
2477                                                    ;STACK AND THE FIRST ERROR CONDITION
2478                                                    ;IS PROBABLY STILL LOCKED UP .
2479  015442  011637  001412          10$:    MOV     (KSP),BADPC     ;SAVE PC AT TIME OF ABORT OR TRAP
```

```
2480  015446  012637  001436                  MOV     (KSP)+,OLDPC    ;SAVE RETURN ADDRESS IN CASE OF LOOP
2481  015452  012637  001440                  MOV     (KSP)+,OLDPS    ;SAVE OLD PSW IN CASE OF LOOP
2482  015456  013737  177572  001404          MOV     MMR0,PMMR0      ;SAVE STATUS REGISTER
2483  015464  013737  177576  001406          MOV     MMR2,PMMR2      ;SAVE VIRTUAL ADDRESS REGISTER
2484  015472  005737  001366                  TST     MMEXP           ;SEE IF ANY M.M. TRAPS WERE EXPECTED
2485  015476  001002                          BNE     5$              ;BRANCH IF ONE WAS EXPECTED
2486  015500  104005                          ERROR   5               ;UNEXPECTED M.M. ABORT OR TRAP
2487  015502  000405                          BR      1$              ;BRANCH TO EXIT
2488  015504  023737  001366  001404  5$:     CMP     MMEXP,PMMR0     ;SEE IF ABORT OR TRAP WAS EXPECTED
2489  015512  001401                          BEQ     1$              ;BRANCH IF CONDITION CORRECT
2490  015514  104006                          ERROR   6               ;ERROR TYPE OUT ITEM 6
2491  015516  042737  177376  177572  1$:     BIC     #177376,@#MMR0  ;CLEAR ALL BITS EXCEPT 0 AND 8
2492  015524  012737  177777  015434          MOV     #-1,MMFLAG      ;RESTORE A NEGATIVE ONE TO FLAG
2493  015532  013746  001440                  MOV     OLDPS,-(KSP)    ;PUSH OLD PSW ONTO STACK
2494  015536  013746  001436                  MOV     OLDPC,-(KSP)    ;PUSH RETURN ADDRESS ON STACK
2495  015542  000006                          RTT                     ;RETURN TO MAIN PROGRAM
2496
2497
2498                                  .SBTTL  TRAP ROUTINES FOR ABORT IN USER MODE
2499                                  ;;*************************************************************************
2500                                  ;*
2501                                  ;*      THESE NEXT THREE SUBROUTINES ARE USED FOR THE TESTS THAT VERIFY
2502                                  ;*      THE VECTOR IS PICKED UP FROM KERNEL SPACE DURING AN ABORT.
2503                                  ;*      'KERVEC' IS WHERE I SHOULD GO SINCE IT IS AT 250 WHICH IS
2504                                  ;*              KERNEL SPACE VIRTUAL 250.
2505                                  ;*      'USEVEC' IS AT 350 WHICH IS USER SPACE VIRTUAL 250
2506                                  ;*
2507                                  ;*      THEY BOTH READ THE PROCESSOR STATUS WHICH IS DIFFERENT FOR EACH
2508                                  ;*      VECTOR AND THEN JUMP TO 'RDMMR0'.  'RDMMR0' READS MMR0
2509                                  ;*      AND MMR2 AND RETURNS TO THE TEST WHERE THEY ARE TESTED.
2510                                  ;*
2511                                  ;;*************************************************************************
2512  015544  013700  177776          KERVEC: MOV     PSW,R0          ;PUT PROCESSOR STATUS INTO R0
2513  015550  012637  001436                  MOV     (KSP)+,OLDPC    ;SAVE RETURN ADDRESS
2514  015554  012637  001440                  MOV     (KSP)+,OLDPS    ;SAVE OLD PROCESSOR STATUS
2515  015560  122700  000340                  CMPB    #340,R0         ;SEE IF CORRECT PSW WAS PICKED UP
2516  015564  001401                          BEQ     1$              ;BRANCH IF PSW IS CORRECT
2517  015566  104072                          ERROR   72              ;WRONG PSW PICKED UP
2518  015570  000137  015612          1$:     JMP     RDMMR0          ;JUMP TO READ MMR0
2519
2520  015574  012637  001436          USEVEC: MOV     (KSP)+,OLDPC    ;SAVE RETURN ADDRESS
2521  015600  012637  001440                  MOV     (KSP)+,OLDPS    ;SAVE OLD PROCESSOR STATUS
2522  015604  013700  177776                  MOV     PSW,R0          ;READ PSW FOR ERROR PRINT OUT
2523  015610  104073                          ERROR   73              ;WRONG VECTOR, POSSIBLE WRONG PSW
2524
2525
2526  015612  013737  177572  001404  RDMMR0: MOV     MMR0,PMMR0      ;READ MMR0 TO BE CHECKED LATER
2527  015620  013737  177576  001406          MOV     MMR2,PMMR2      ;READ MMR2 FOR POSSIBLE ERROR REPORT
2528  015626  013746  001440                  MOV     OLDPS,-(KSP)    ;PUSH OLD PROCESSOR STATUS ON STACK
2529  015632  013746  001436                  MOV     OLDPC,-(KSP)    ;PUSH RETURN ADDRESS ON STACK
2530  015636  000006                          RTT                     ;RETURN TO TEST TO CHECK PMMR0
2531
2532
```

```
                                        .SBTTL
                                        .SBTTL  ************ ENTRY POINT 1 --- STARTING ADDRESS 200 ************
                                        .SBTTL  ********** TEST CODE STARTS AT ADDRESS 20000 (PAGE 1) **********
                    020000              .=20000
020000  012737  000000  001362  STRT1:  MOV     #0,FSTTST       ;LOAD INDEX TO START TABLE
020006  000427                  STRT2:  BR      START           ;BRANCH TO STARTING ADDRESS OF PROGRAM
020010  012737  000002  001362          MOV     #2,FSTTST       ;LOAD INDEX TO START TABLE
020016  000423                  STRT3:  BR      START           ;BRANCH TO STARTING ADDRESS OF PROGRAM
020020  012737  000004  001362          MOV     #4,FSTTST       ;LOAD INDEX TO START TABLE
020026  000417                  STRT4:  BR      START           ;BRANCH TO STARTING ADDRESS OF PROGRAM
020030  012737  000006  001362          MOV     #6,FSTTST       ;LOAD INDEX TO START TABLE
020036  000413                  STRT5:  BR      START           ;BRANCH TO STARTING ADDRESS OF PROGRAM
020040  012737  000010  001362          MOV     #10,FSTTST      ;LOAD INDEX TO START TABLE
020046  000407                  STRT6:  BR      START           ;BRANCH TO STARTING ADDRESS OF PROGRAM
020050  012737  000012  001362          MOV     #12,FSTTST      ;LOAD INDEX TO START TABLE
020056  000403                  STRT7:  BR      START           ;BRANCH TO STARTING ADDRESS OF PROGRAM
020060  012737  000014  001362          MOV     #14,FSTTST      ;LOAD INDEX TO START TABLE

                                ;*      NOTE:   THE FIRST THING THAT IS DONE IS TO SEE IF THE
                                ;*              11/40 OR THE 11/6X MEMORY MGMT. IS BEING TESTED
                                ;*              AND IF ON AN 11/40 IF THERE IS A STACK LIMIT
                                ;*              REGISTER.  THE 11/6X "MED" INSTRUCTION IS USED
                                ;*              TO DETERMINE PROCESSOR TYPE AND THE STACK LIMIT
                                ;*              REG. IS CHECKED FOR TIMEOUT.
                                ;*
                                ;*              IF A "MED" DOES NOT EXECUTE PROPERLY ON AN 11/6X
                                ;*              OR IF THE STACK LIMIT REG. TIMES OUT ON AN 11/40 -
                                ;*              ERRONEOUS DECISIONS WILL BE MADE DURING TESTING
                                ;*              WHICH WILL RESULT IN INCOMPLETE TESTING.

020066  012706  001100  START:  MOV     #KERSTK,KSP     ;SETUP STACK POINTER
020072  012737  020112  000010          MOV     #1$,@#10        ;SETUP ILLEGAL INST. VECTOR
020100  105037  001450          CLRB    FORTY           ;CLEAR 11/40 "FLAG LOC."
020104  076600                  MED                     ;TRY 11/6X INSTRUCTION
020106  000100                  RLJAM                   ;WILL TRAP TO 1$ IF ON AN 11/40
020110  000410                  BR      3$              ;WILL BRANCH TO SETUP IF ON 11/6X
020112  062706  000004  1$:     ADD     #4,KSP          ;CLEAN UP STACK IF ON 11/40 AND
020116  105237  001450          INCB    FORTY           ;SET 11/40 "FLAG LOC."
020122  012737  000012  000010          MOV     #12,@#10        ;RESTORE TRAP CATCHER TO LOC. 10
020130  000403                  BR      4$              ;BRANCH TO SETUP
020132  012777  000014  161212  3$:     MOV     #14,@CONTRL     ;TURN 11/6X CACHE OFF FOR FIRST PASS
020140                          4$:
                                .SBTTL  INITIALIZE THE COMMON TAGS
                                ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
020140  012706  001100          MOV     #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
020144  005026                  CLR     (R6)+           ;;CLEAR MEMORY LOCATION
020146  022706  001140          CMP     #$WR,R6 ;;DONE?
020152  001374                  BNE     .-6             ;;LOOP BACK IF NO
020154  012706  001100          MOV     #STACK,SP       ;;SETUP THE STACK POINTER
                                ;;INITIALIZE A FEW VECTORS
020160  012737  043004  000020          MOV     #$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
020166  012737  000340  000022          MOV     #340,@#IOTVEC+2 ;;LEVEL 7
020174  012737  043300  000030          MOV     #$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
020202  012737  000340  000032          MOV     #340,@#EMTVEC+2 ;;LEVEL 7
```

```
                        INITIALIZE THE COMMON TAGS

  2589  020210  012737  042340  000034          MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
  2590  020216  012737  000340  000036          MOV     #340,@#TRAPVEC+2;LEVEL 7
  2591  020224  012737  042420  000024          MOV     #$PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
  2592  020232  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;LEVEL 7
  2593  020240  013737  040610  040602          MOV     $ENDCT,$EOPCT   ;;SETUP END-OF-PROGRAM COUNTER
  2594  020246  005037  001212                  CLR     $TIMES          ;;INITIALIZE NUMBER OF ITERATIONS
  2595  020252  005037  001214                  CLR     $ESCAPE         ;;CLEAR THE ESCAPE ON ERROR ADDRESS
  2596  020256  112737  000001  001115          MOVB    #1,$ERMAX       ;;ALLOW ONE ERROR PER TEST
  2597                                  ;;INITIALIZE THE "T-BIT" TRAP VECTOR. THEN LOAD LOCATION "$RTRN", IN
  2598                                  ;;THE "END-OF-PASS" ($EOP) ROUTINE, WITH A "RTI" OR "RTT".
  2599  020264  012737  041026  000014          MOV     #$RTRN,@#TBITVEC ;;SET "T" BIT VECTOR TO $RTRN
  2600  020272  012737  000340  000016          MOV     #340,@#TBITVEC+2 ;;LEVEL 7
  2601  020300  012737  000002  041026          MOV     #RTI,$RTRN      ;;SET $RTRN TO A RTI
  2602  020306  012737  020334  000010          MOV     #65$,@#RESVEC   ;;TRY TO DO A RTT
  2603  020314  005046                          CLR     -(SP)           ;;DUMMY PS
  2604  020316  012746  020324                  MOV     #64$,-(SP)      ;;AND PC
  2605  020322  000006                          RTT                     ;;TRY THE RTT
  2606  020324  012737  000006  041026  64$:    MOV     #RTT,$RTRN      ;;RTT IS LEGAL--SET $RTRN TO A RTT
  2607  020332  000402                          BR      66$
  2608  020334  062706  000010          65$:    ADD     #10,SP          ;;RTT ILLEGAL--CLEAN OFF THE STACK
  2609  020340  012737  000012  000010  66$:    MOV     #RESVEC+2,@#RESVEC ;;RESTORE TRAP CATCHER
  2610  020346  005037  041034                  CLR     $TBIT           ;;CLEAR "T" BIT SWITCH
  2611  020352  012737  020352  001106          MOV     #.,$LPADR       ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
  2612  020360  012737  020360  001110          MOV     #.,$LPERR       ;;SETUP THE ERROR LOOP ADDRESS
  2613                                  ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
  2614                                  ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
  2615  020366  013746  000004                  MOV     @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
  2616  020372  012737  020426  000004          MOV     #67$,@#ERRVEC   ;;SET UP ERROR VECTOR
  2617  020400  012737  177570  001140          MOV     #DSWR,SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
  2618  020406  012737  177570  001142          MOV     #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
  2619  020414  022777  177777  160516          CMP     #-1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
  2620  020422  001012                          BNE     69$             ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
  2621                                                                  ;;AND  THE HARDWARE SWR IS NOT = -1
  2622  020424  000403                          BR      68$             ;;BRANCH IF NO TIMEOUT
  2623  020426  012716  020434          67$:    MOV     #68$,(SP)       ;;SET UP FOR TRAP RETURN
  2624  020432  000002                          RTI
  2625  020434  012737  000176  001140  68$:    MOV     #SWREG,SWR      ;;POINT TO SOFTWARE SWR
  2626  020442  012737  000174  001142          MOV     #DISPREG,DISPLAY
  2627  020450  012637  000004          69$:    MOV     (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR
  2628
  2629  020454  005037  001234                  CLR     $PASS           ;;CLEAR PASS COUNT
  2630  020460  132737  000200  001247          BITB    #APTSIZE,$ENVM  ;;TEST USER SIZE UNDER APT
  2631  020466  001403                          BEQ     70$             ;;YES,USE NON-APT SWITCH
  2632  020470  012737  001250  001140          MOV     #$SWREG,SWR     ;;NO,USE APT SWITCH REGISTER
  2633  020476                          70$:
  2634                                  .SBTTL  TYPE PROGRAM NAME
  2635                                  ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
  2636  020476  005227  177777                  INC     #-1             ;;FIRST TIME?
  2637  020502  001035                          BNE     71$             ;;BRANCH IF NO
  2638  020504  022737  040762  000042          CMP     #$ENDAD,@#42    ;;ACT-11?
  2639  020512  001431                          BEQ     71$             ;;BRANCH IF YES
  2640  020514  104401  020522                  TYPE    ,72$            ;;TYPE ASCIZ STRING
  2641  020520  000426                          BR      71$             ;;GET OVER THE ASCIZ
  2642                                  ;;72$:  .ASCIZ  <CRLF>?MAINDEC-11-DQKTA-A 11/6X MEM. MGMT. DIAG.?<CRLF>
  2643  020576                          71$:
  2644  020576  012737  020576  001110          MOV     #.,$LPERR       ;INITIALIZE LOOP ON ERROR PTR.
```

```
2645  020604  012737  020604  001106          MOV     #.,$LPADR       ;INITIALIZE LOOP ADDRESS PTR.
2646
2647
2648  020612  005737  001234          LOOP:   TST     $PASS           ;IS THIS THE FIRST PASS?
2649  020616  001406                          BEQ     5$              ;BRANCH IF FIRST PASS
2650  020620  105737  001450                  TSTB    FORTY           ;EXECUTING ON AN 11/40?
2651  020624  001003                          BNE     5$              ;BRANCH IF YES
2652  020626  012777  000200  160516          MOV     #200,@CONTRL    ;OTHERWISE TURN CACHE ON AND SET THE
2653                                                                  ;PARITY ABORT BIT
2654  020634  005037  177572          5$:     CLR     MMR0            ;GET INTO 16 BIT MODE ON SECOND PASS
2655  020640  012700  177777                  MOV     #-1,R0          ;PUT NEG ONE IN R0 TO INITIALIZE FLAGS
2656  020644  010037  015164                  MOV     R0,CPFLAG       ;INITIALIZE CPU ERROR FLAG
2657  020650  010037  015254                  MOV     R0,PAFLAG       ;INITIALIZE PARITY ERROR FLAG
2658  020654  010037  015434                  MOV     R0,MMFLAG       ;INITIALIZE MEMORY MANAGEMENT TRAP FLAG
2659  020660  005037  001364                  CLR     CPUEXP          ;NOT EXPECTING ANY CPU TRAPS
2660  020664  005037  001366                  CLR     MMEXP           ;NOT EXPECTING ANY MEMORY MANAGEMENT TRAPS
2661  020670  012737  015162  000004          MOV     #CPUER,ERRVEC   ;LOAD ADDRESS OF CPU TRAP ROUTINE
2662  020676  012737  000340  000006          MOV     #340,ERRVEC+2   ;SET PRIORITY LEVEL 7
2663  020704  012737  015252  000114          MOV     #MEMER,CACHVEC  ;LOAD ADDRESS OF PARITY TRAP ROUTINE
2664  020712  012737  000340  000116          MOV     #340,CACHVEC+2  ;SET PRIORITY LEVEL 7
2665  020720  012737  015432  000250          MOV     #MMTRAP,MMVEC   ;LOAD ADDRESS OF MEMORY MANAGEMENT TRAP
2666  020726  012737  000340  000252          MOV     #340,MMVEC+2    ;SET PRIORITY LEVEL 7
2667  020734  004737  044202                  JSR     PC,CLEANUP      ;INITIALIZE ALL ERROR LOCATIONS
2668  020740  012706  001100                  MOV     #KERSTK,KSP     ;SET UP KERNEL STACK POINTER
2669  020744  013700  001362                  MOV     FSTTST,R0       ;LOAD START TABLE INDEX
2670  020750  001412                          BEQ     TST1            ;START WITH TEST ONE IF ZERO
2671  020752  012737  140000  177776          MOV     #140000,PSW     ;GO TO USER MODE
2672  020760  012706  000700                  MOV     #USESTK,USP     ;SET UP USER STACK POINTER
2673  020764  012737  000340  177776          MOV     #340,PSW        ;GO TO KERNEL MODE PRIORITY LEVEL 7
2674  020772  000170  001462                  JMP     @STRTAB(R0)     ;JUMP TO CORRECT ENTRY POINT
2675
2676
2677                                  ;*      THIS FIRST GROUP OF TESTS IS FOR TESTING THE ADDRESS DECODE
2678                                  ;*      LOGIC FOR THE INTERNAL REGISTERS AND TO MAKE SOME
2679                                  ;*      DETERMINATION ABOUT THE RESPONDING ADDRESSES.  IF AN
2680                                  ;*      ADDRESS DOES NOT FUNCTION AS THE CORRESPONDING REGISTER SHOULD
2681                                  ;*      AN ERROR WILL BE FLAGGED.  THE MULTIPLEXERS AND INTERNAL BUS
2682                                  ;*      DRIVERS ARE ALSO UTILIZED IN THESE TESTS AND COULD BE
2683                                  ;*      THE SOURCE OF ANY PROBLEMS ENCOUNTERED.
2684
2685
2686                                  ;;*********************************************************************
2687                                  ;*TEST 1        TRY TO READ ALL CPU REGISTERS
2688                                  ;*
2689                                  ;*      THIS TEST ENSURES THAT SOMETHING RESPONDS TO SOME INTERNAL ADDRESSES.
2690                                  ;*      THIS PROCESSOR STATUS WORD AND STACK LIMIT REG. (IF
2691                                  ;*      AVAILABLE) ARE TESTED IF EXECUTING ON AN 11/40, THE CPU
2692                                  ;*      ERROR REG. 2 MICROBREAK REG. ARE ALSO CHECKED IF ON AN 11/6X.
2693                                  ;*      IF A TRAP TO 'ERRVEC' (004) OCCURS IT IS ASSUMED
2694                                  ;*      THAT A REGISTER HAS TIMED OUT, AND AN ERROR IS REPORTED.
2695                                  ;*
2696                                  ;*      ERRORS THAT OCCUR IN THIS TEST ARE REPORTED FROM AN AREA
2697                                  ;*      AT THE END OF THIS TEST, STARTING AT "50$".
2698                                  ;*
2699                                  ;;*********************************************************************
2700  020776                          TST1:
```

```
2701  020776  012737  021026  001110          MOV     #20$,$LPERR     ;SET LOOP ON ERROR POINTER TO 20$
2702  021004  012737  000001  001102          MOV     #1,$TSTNM       ;LOAD TEST NUMBER INTO MEMORY
2703  021012  013777  001102  160122          MOV     $TSTNM,@DISPLAY ;DISPLAY TEST NUMBER FOR THIS TEST
2704  021020  012737  021130  001446          MOV     #TST2,NXTTST    ;SAVE STARTING ADDRESS OF NEXT TEST
2705                                                                  ;FOR ESCAPE ON PARITY ERRORS
2706  021026  012737  021124  000004  20$:     MOV     #50$,ERRVEC     ;SET TIME OUT VECTOR TO SPECIAL ROUTINE
2707  021034  012706  001100                  MOV     #1100,KSP       ;SET KERNEL STACK POINTER TO 1100
2708  021040  012700  177776          1$:     MOV     #PSW,R0         ;SAVE ADDRESS IN CASE OF TIMEOUT
2709  021044  005737  177776                  TST     @#PSW           ;TRY TO READ PROCESSOR STATUS WORD
2710  021050  012700  177774                  MOV     #STKLMT,R0      ;SAVE ADDRESS IN CASE OF TIMEOUT
2711  021054  005737  177774                  TST     @#STKLMT        ;TRY TO READ STACK LIMIT REG.
2712
2713  021060  105737  001450                  TSTB    FORTY           ;EXECUTING ON AN 11/40?
2714  021064  001010                          BNE     2$              ;BRANCH IF YES
2715  021066  013700  001354                  MOV     CPUERR,R0       ;SAVE ADDRESS IN CASE OF TIMEOUT
2716  021072  005777  160256                  TST     @CPUERR         ;TRY TO READ CPU ERROR REG.
2717  021076  012700  177770                  MOV     #177770,R0      ;SAVE ADDRESS IN CASE OF TIMEOUT
2718  021102  005737  177770                  TST     @#177770        ;TRY TO READ MICROBREAK REG.
2719
2720  021106                          2$:
2721  021106  012737  021026  001110          MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
2722  021114  012737  015162  000004          MOV     #CPUER,ERRVEC   ;SET UP C.P.U. ERROR VECTOR
2723  021122  000402                          BR      TST2            ;;TEST OVER BRANCH TO NEXT TEST
2724
2725  021124  104020                  50$:     ERROR   20              ;CPU REGISTER TIMED OUT
2726  021126  000006                          RTT                     ;GO BACK AND FINISH TEST
2727
2728
2729
2730
2731
2732
2733
2734                            ;*****************************************************************
2735                            ;*TEST 2        STACK LIMIT REGISTER
2736                            ;*
2737                            ;*      THE STACK LIMIT REGISTER IS A HIGH(ODD) BYTE ONLY REGISTER, SO THIS
2738                            ;*      TEST TRIES TO LOAD BOTH BYTES OF ADDRESS 177774 AND CHECKS
2739                            ;*      THAT ONLY THE HIGH(ODD) BYTE IS LOADED.  THE LOW(EVEN) BYTE WILL ALWAYS
2740                            ;*      BE READ AS ZERO.
2741                            ;*      THIS TEST WILL ALSO BYTE ADDRESS THE STACK LIMIT
2742                            ;*      REGISTER USING ADDR. 177775 AND WRITE/CLEAR
2743                            ;*      EACH BIT OF THE HIGH(ODD) BYTE.
2744                            ;*      A RESET IS ALSO EXECUTED TO SEE THAT IT WILL
2745                            ;*      CLEAR THE STACK LIMIT REGISTER(DUE TO MICROCODE
2746                            ;*      FOR RESET)
2747                            ;*
2748                            ;*      THIS TEST IS SKIPPED IF EXECUTING ON AN 11/40 THAT DOES
2749                            ;*      NOT HAVE A STACK LIMIT REG.
2750                            ;*
2751                            ;*****************************************************************
2752  021130                  TST2:
2753  021130  000004                  SCOPE
2754  021132  012737  021340  001446          MOV     #TST3,NXTTST    ;SAVE STARTING ADDRESS OF NEXT
2755                                                                  ;TEST FOR ESCAPE ON PARITY ERRORS
2756  021140  104410                          TBITO                   ;MAKE SURE T-BIT IS OFF FOR THIS TEST
```

```
2757  021142  012737  021154  001110  20$:    MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
2758  021150  012700  177774                  MOV     #177774,R0      ;LOAD ADDRESS OF STACK LIMIT REGISTER
2759  021154  012710  000777          1$:     MOV     #777,(R0)       ;ONLY HIGH(ODD) BYTE SHOULD BE LOADED
2760  021160  011001                          MOV     (R0),R1         ;READ STACK LIMIT REGISTER
2761  021162  005010                          CLR     (R0)            ;LEAVE STACK LIMIT REGISTER CLEAR
2762  021164  012702  000400                  MOV     #400,R2         ;LOAD EXPECTED DATA INTO R2
2763  021170  020102                          CMP     R1,R2           ;DID YOU REFERENCE THE STACK LIMIT REG
2764  021172  001401                          BEQ     2$              ;BRANCH IF IT WAS STACK LIMIT
2765  021174  104026                          ERROR   26              ;NOT STACK LIMIT REGISTER
2766  021176  005200          2$:             INC     R0              ;CHANGE ADDRESS TO REFERENCE HIGH(ODD) BYTE
2767  021200  012737  021206  001110          MOV     #3$,$LPERR      ;SET LOOP ON ERROR POINTER TO 3$
2768  021206  112710  000001          3$:     MOVB    #1,(R0)         ;SHOULD ONLY LOAD HIGH(ODD) BYTE
2769  021212  005300                          DEC     R0              ;CHANGE ADDRESS TO REFERENCE WORD
2770  021214  011001                          MOV     (R0),R1         ;READ STACK LIMIT REGISTER
2771  021216  005010                          CLR     (R0)            ;LEAVE STACK LIMIT REGISTER CLEAR
2772  021220  012702  000400                  MOV     #400,R2         ;LOAD EXPECTED DATA INTO R2
2773  021224  020102                          CMP     R1,R2           ;DID YOU REFERENCE HIGH(ODD) BYTE
2774  021226  001403                          BEQ     4$              ;BRANCH IF DATA MATCHES
2775  021230  005200                          INC     R0              ;CHANGE ADDRESS TO ODD BYTE FOR ERROR
2776  021232  104025                          ERROR   25              ;WRONG DATA BACK FROM STACK LIMIT REG
2777  021234  005300                          DEC     R0              ;CHANGE ADDRESS BACK TO WORD ADDRESS
2778  021236  012737  021250  001110  4$:     MOV     #5$,$LPERR      ;SET LOOP ON ERROR POINTER TO 5$
2779  021244  012702  000400                  MOV     #400,R2         ;LOAD TEST PATTERN INTO R2
2780  021250  010210          5$:             MOV     R2,(R0)         ;LOAD TEST PATTERN INTO STACK LIMIT REG.
2781  021252  011001                          MOV     (R0),R1         ;READ STACK LIMIT REGISTER
2782  021254  020201                          CMP     R2,R1           ;WAS CORRECT DATA READ
2783  021256  001402                          BEQ     6$              ;BRANCH IF DATA MATCHES
2784  021260  005010                          CLR     (R0)            ;CLEAR STK. LMT. REG. FOR ERROR CALL
2785  021262  104025                          ERROR   25              ;WRONG DATA BACK FROM STACK LIMIT REG.
2786  021264  012737  021264  001110  6$:     MOV     #6$,$LPERR      ;SET LOOP ON ERROR POINTER TO 6$
2787  021272  005010                          CLR     (R0)            ;TRY TO CLEAR THE STACK LIMIT REG
2788  021274  011001                          MOV     (R0),R1         ;READ THE STACK LIMIT REGISTER
2789  021276  001401                          BEQ     7$              ;BRANCH IF STACK LIMIT REG WAS CLEARED
2790  021300  104021                          ERROR   21              ;STACK LIMIT REG WOULD NOT CLEAR
2791  021302  006302          7$:             ASL     R2              ;CHANGE TEST DATA TO TEST NEXT BIT
2792  021304  012737  021250  001110          MOV     #5$,$LPERR      ;SET LOOP ON ERROR POINTER TO 5$
2793  021312  103356                          BCC     5$              ;BRANCH UNTIL BITS ARE ALL TESTED
2794  021314  012710  177400                  MOV     #177400,(R0)    ;LOAD STACK LIMIT REG. WITH ALL 1'S
2795  021320  000005                          RESET                   ;TRY TO CLEAR WITH A RESET INST.
2796  021322  011001                          MOV     (R0),R1         ;READ THE STACK LIMIT REG.
2797  021324  001402                          BEQ     8$              ;BRANCH IF WAS CLEARED
2798  021326  005010                          CLR     (R0)            ;CLEAR STK. LMT. REG. FOR ERROR CALL
2799  021330  104021                          ERROR   21              ;STACK LIMIT REG. WOULD NOT CLEAR
2800  021332  012737  021142  001110  8$:     MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
2801
2802
2803
2804                                  ;;*****************************************************************
2805                                  ;*TEST 3        PROCESSOR STATUS WORD
2806                                  ;*
2807                                  ;*      THIS TEST SETS THE PRIORITY LEVEL TO 7 AND CLEARS THE
2808                                  ;*      CONDITION CODES AND CHECKS TO SEE THAT ADDRESS 177776
2809                                  ;*      HAS 340 IN ITS LOW BYTE.
2810                                  ;*
2811                                  ;*****************************************************************
2812  021340                          TST3:
```

```
2813  021340  000004                          SCOPE
2814  021342  012737  021430  001446          MOV     #TST4,NXTTST    ;SAVE STARTING ADDRESS OF NEXT
2815                                                                  ;TEST FOR ESCAPE ON PARITY ERRORS
2816  021350  104411                          TBITR                   ;RESTORE THE T-BIT IF IT WAS ON
2817                                                                  ;BEFORE THE LAST TEST
2818  021352  012737  021360  001106          MOV     #20$,$LPADR     ;SET LOOP ADDRESS POINTER TO 20$
2819                                                                  ;FOR ITERATION PASS
2820  021360  012737  021376  001110  20$:     MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
2821  021366  012700  177776                  MOV     #177776,R0      ;LOAD ADDRESS OF P.S.W. INTO R0
2822  021372  012702  000340                  MOV     #340,R2         ;LOAD EXPECTED DATA INTO R2
2823  021376  005010                  1$:     CLR     (R0)            ;CLEAR THE PSW
2824  021400  052710  000340                  BIS     #340,(R0)       ;SET PRIORITY TO LEVEL SEVEN
2825  021404  000257                          CCC                     ;CLEAR ALL CONDITION CODES
2826  021406  111001                          MOVB    (R0),R1         ;LOAD LOWER BYTE OF P.S.W. INTO R1
2827  021410  042701  177420                  BIC     #177420,R1      ;CLEAR "REG. SIGN EXTEND" AND T-BIT
2828  021414  020201                          CMP     R2,R1           ;LOWER BYTE OF PSW SHOULD BE 340
2829  021416  001401                          BEQ     2$              ;BRANCH IF PSW IS CORRECT
2830  021420  104023                          ERROR   23              ;MUST HAVE READ SOME OTHER REGISTER
2831  021422  012737  021360  001110  2$:     MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
2832
2833
2834                                  ;;**********************************************************************
2835                                  ;*TEST 4        SET UP THE STACK POINTERS FOR REST OF TESTS
2836                                  ;*
2837                                  ;*      THIS TEST IS USED TO SET THE KERNEL STACK POINTER AT
2838                                  ;*      1100 AND THE USER STACK POINTER AT 700.  IT THEN RETURNS
2839                                  ;*      TO KERNEL MODE AND VERIFIES THAT THE KERNEL STACK
2840                                  ;*      POINTER IS STILL AT 1100.
2841                                  ;*
2842                                  ;;**********************************************************************
2843  021430                          TST4:
2844  021430  000004                          SCOPE
2845  021432  012737  021500  001446          MOV     #TST5,NXTTST    ;SAVE STARTING ADDRESS OF NEXT
2846                                                                  ;TEST FOR ESCAPE ON PARITY ERRORS
2847  021440  005037  177776                  CLR     PSW             ;GET TO KERNEL MODE, PRIORITY 0
2848  021444  012706  001100                  MOV     #KERSTK,KSP     ;SETUP KERNEL STACK POINTER
2849  021450  012737  140000  177776          MOV     #140000,PSW     ;GET INTO USER MODE
2850  021456  012706  000700                  MOV     #USESTK,USP     ;SET UP USER STACK POINTER
2851  021462  005037  177776                  CLR     PSW             ;GET BACK INTO KERNEL MODE
2852  021466  010600                          MOV     KSP,R0          ;READ KSRNEL STACK POINTER
2853  021470  022700  001100                  CMP     #KERSTK,R0      ;SEE IF KERNEL STACK IS STILL 1100
2854  021474  001401                          BEQ     TST5            ;;BRANCH IF KERNEL STACK IS OKAY
2855  021476  104027                          ERROR   27              ;KERNEL STACK POINTER IS NOT RIGHT
2856
2857
2858                                  .SBTTL  ************ ENTRY POINT 2 --- STARTING ADDRESS 204 ************
2859                                  .SBTTL  ** TEST READ/WRITE BITS IN MEMORY MANAGEMENT STATUS REGISTERS **
2860                                  ;*
2861                                  ;*      THIS GROUP OF TESTS EXERCISES ALL OF THE READ/WRITE BITS
2862                                  ;*      IN MMR0, AND TRIES A FEW VIRTUAL ADDRESSES IN MMR2.
2863                                  ;*      THESE TESTS SHOW THAT ONCE THE MMR LOCK UP SIGNAL GOES LOW
2864                                  ;*      MMR2 STOPS TRACKING THE C.P.U. OPERATIONS.
2865                                  ;*
2866
2867
2868                                  ;;**********************************************************************
```

```
                                        ;*TEST 5        BIT TEST OF MEMORY MANAGEMENT REGISTER 0
                                        ;*
                                        ;*      THIS TEST TRIES TO SET AND CLEAR BITS <15:13> OF
                                        ;*      MMR0.  'INIT' IS ISSUED TO SEE THAT IT WILL CLEAR THESE BITS.
                                        ;*      THE OTHER BITS OF MMR0 ARE CLOCKED ONLY ON MEMORY
                                        ;*      MANAGEMENT ERROR CONDITIONS IF RELOCATION IS ENABLED.
                                        ;*      BIT <00> ENABLES FULL RELOCATION AND BIT <08> ENABLES
                                        ;*      RELOCATION ON THE DESTINATION CYCLE ONLY.  THESE BITS WILL
                                        ;*      BE TESTED IN A LATER TEST.
                                        ;*
                                        ;;*********************************************************************
021500                                  TST5:
021500  000004                                  SCOPE
021502  012737  021666  001446                 MOV     #TST6,NXTTST    ;SAVE STARTING ADDRESS OF NEXT
                                                                        ;TEST FOR ESCAPE ON PARITY ERRORS
021510                                  ENTPT2:
021510  012737  021532  001110                 MOV     #20$,$LPERR     ;SET LOOP ON ERROR POINTER TO 20$
021516  012737  000005  001102                 MOV     #5,$TSTNM       ;LOAD TEST NUMBER INTO MEMORY
021524  013777  001102  157410                 MOV     $TSTNM,@DISPLAY ;DISPLAY TEST NUMBER FOR THIS TEST
021532  012700  177572          20$:    MOV     #MMR0,R0        ;PUT ADDRESS OF MMR0 IN R0
021536  012710  160000                  MOV     #160000,(R0)    ;LOAD WRITABLE BITS IN MMR0
021542  000005                          RESET                   ;'INIT', SHOULD CLEAR ALL BITS OF MMR0
021544  011001                          MOV     (R0),R1         ;READ MMR0
021546  001401                          BEQ     1$              ;BRANCH IF MMR0 IS CLEAR
021550  104007                          ERROR   7               ;MMR0 WON'T CLEAR
021552  012737  021560  001110  1$:     MOV     #11$,$LPERR     ;SET LOOP ON ERROR POINTER TO 11$
021560  012710  160000          11$:    MOV     #160000,(R0)    ;SET BITS <15:13> OF MMR0
021564  011001                          MOV     (R0),R1         ;READ MMR0
021566  022701  160000                  CMP     #160000,R1      ;SEE IF BITS <15:13> ARE SET
021572  001401                          BEQ     2$              ;BRANCH IF CORRECT BITS ARE SET
021574  104010                          ERROR   10              ;CAN'T SET 160000 IN MMR0
021576  012737  021604  001110  2$:     MOV     #12$,$LPERR     ;SET LOOP ON ERROR POINTER TO 12$
021604  005010          12$:    CLR     (R0)            ;CLEAR UPPER 3 BITS OF MMR0
021606  011001                          MOV     (R0),R1         ;READ MMR0
021610  001401                          BEQ     3$              ;BRANCH IF MMR0 IS CLEAR
021612  104007                          ERROR   7               ;MMR0 WON'T CLEAR
021614  012737  021632  001110  3$:     MOV     #5$,$LPERR      ;SET LOOP ON ERROR POINTER TO 5$
021622  012702  100000                  MOV     #BIT15,R2       ;SET BIT IN R2 TO FLOAT THRU MMR0
021626  012703  000003                  MOV     #3,R3           ;PUT LOOP COUNT IN R3
021632  010210          5$:     MOV     R2,(R0)         ;LOAD R2 INTO MMR0
021634  011001                          MOV     (R0),R1         ;READ MMR0 IN R1
021636  005010                          CLR     (R0)            ;CLEAR MMR0
021640  010204                          MOV     R2,R4           ;PUT PATTERN IN R4
021642  042704  017777                  BIC     #017777,R4      ;MASK OFF BITS <12:00>
021646  020401                          CMP     R4,R1           ;COMPARE PATTERN WITH MMR0
021650  001401                          BEQ     4$              ;BRANCH IF DATA MATCHES
021652  104011                          ERROR   11              ;GOT WRONG DATA FROM MMR0
021654  006002          4$:     ROR     R2              ;SHIFT BIT TO RIGHT
021656  077313                          SOB     R3,5$           ;LOOP BACK TWICE
021660  012737  021532  001110          MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST


                                        ;;*********************************************************************
                                        ;*TEST 6        SEE THAT MEMORY MANAGEMENT REGISTER 1 READS 0'S
                                        ;*
```

```
                                        T6      SEE THAT MEMORY MANAGEMENT REGISTER 1 READS 0'S

                                        ;*      THIS TEST WILL ENSURE THAT ALTHOUGH MEMORY MANAGEMENT
                                        ;*      REGISTER #1 (777574) IS NON-EXSISTENT, THAT ADDRESS
                                        ;*      WILL RESPOND AND READ AS THOUGH THERE WAS A REGISTER
                                        ;*      THERE AND THAT IT READS AS ALL ZEROES.
                                        ;*
                                        ;**********************************************************************
021666                                  TST6:
021666  000004                                SCOPE
021670  012737  021716  001446                MOV     #TST7,NXTTST    ;SAVE STARTING ADDRESS OF NEXT
                                                                        ;TEST FOR ESCAPE ON PARITY ERRORS

021676  012700  177574          20$:    MOV     #MMR1,R0        ;PUT ADDRESS OF MMR1 IN R0
021702  012701  177777                  MOV     #-1,R1          ;LOAD R1 WITH ALL ONES
021706  011001                          MOV     (R0),R1         ;READ MMR1 INTO R1
021710  005701                          TST     R1              ;DID MMR1 READ AS ALL ZEROES?
021712  001401                          BEQ     1$              ;BRANCH IF YES
021714  104024                          ERROR   24              ;MMR1 DID NOT READ ALL ZEROES
021716                          1$:



                                        ;**********************************************************************
                                        ;*TEST 7        BIT TEST OF MEMORY MANAGEMENT REGISTER 2
                                        ;*
                                        ;*      HERE MMR2 WILL BE READ SEVERAL DIFFERENT TIMES TO
                                        ;*      SEE THAT IT IS TRACKING PROPERLY.  THEN IT WILL BE
                                        ;*      LOCKED UP AND READ IT TO SEE THAT IT DOES NOT CHANGE AFTER
                                        ;*      IT IS ONCE LOCKED UP.  NOT ALL BITS WILL BE TESTED IN THIS
                                        ;*      TEST BUT A LATER TEST RUNS A COUNT PATTERN THROUGH MMR2.  THIS
                                        ;*      CANNOT BE DONE HERE SINCE IT REQUIRES THAT THE ABORT LOGIC IS
                                        ;*      FUNCTIONING PROPERLY.
                                        ;*
                                        ;**********************************************************************
021716                                  TST7:
021716  000004                                SCOPE
021720  012737  022062  001446                MOV     #TST10,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
                                                                        ;TEST FOR ESCAPE ON PARITY ERRORS
021726  012700  177572          20$:    MOV     #MMR0,R0        ;PUT ADDRESS OF MMR0 IN R0
021732  012701  177576                  MOV     #MMR2,R1        ;PUT ADDRESS OF MMR2 IN R1
021736  012737  021750  001110          MOV     #21$,$LPERR     ;SET LOOP ON ERROR POINTER TO 21$
021744  005037  177572                  CLR     @#MMR0          ;MAKE SURE THAT MMR0 IS CLEAR
021750  011102                  21$:    MOV     (R1),R2         ;READ MMR2 TO R2
021752  012703  021750                  MOV     #21$,R3         ;PUT ADDRESS OF LAST INST IN R3
021756  020203                          CMP     R2,R3           ;SEE IF MMR2 HELD CORRECT ADDRESS
021760  001401                          BEQ     1$              ;BRANCH IF DATA MATCHES
021762  104012                          ERROR   12              ;MMR2 DID NOT TRACK PROPERLY
021764  012737  021772  001110  1$:     MOV     #22$,$LPERR     ;SET LOOP ON ERROR POINTER TO 22$
021772  013702  177576          22$:    MOV     @#MMR2,R2       ;READ MMR2 TO R2
021776  012703  021772                  MOV     #22$,R3         ;PUT ADDRESS OF LAST INST IN R3
022002  020203                          CMP     R2,R3           ;SEE IF MMR2 HELD CORRECT ADDRESS
022004  001401                          BEQ     2$              ;BRANCH IF DATA MATCHES
022006  104012                          ERROR   12              ;MMR2 DID NOT TRACK PROPERLY
022010  012737  022016  001110  2$:     MOV     #23$,$LPERR     ;SET LOOP ON ERROR POINTER TO 23$
022016  012710  040000          23$:    MOV     #40000,(R0)     ;LOCK UP MMR2
022022  011102                          MOV     (R1),R2         ;READ MMR2 INTO R2
022024  012703  022016                  MOV     #23$,R3         ;PUT LOCKING INST'S ADDRESS IN R3
```

```
2981  022030  020203                        CMP     R2,R3           ;SEE IF MMR2 HOLDS RIGHT ADDRESS
2982  022032  001401                        BEQ     3$              ;BRANCH IF DATA MATCHES
2983  022034  104012                        ERROR   12              ;MMR2 DID NOT TRACK PROPERLY
2984  022036  012737  022044  001110  3$:   MOV     #14$,$LPERR     ;SET LOOP ON ERROR POINTER TO 14$
2985  022044  000005                 14$:   RESET                   ;ISSUE INIT TO CLEAR MMR2
2986  022046  011102                 25$:   MOV     (R1),R2         ;READ MMR2 INTO R2
2987  022050  012703  022046               MOV     #25$,R3         ;PUT ADDRESS OF LAST INST IN R3
2988  022054  020203                        CMP     R2,R3           ;SEE IF MMR2 IS STILL TRACKING
2989  022056  001401                        BEQ     TST10           ;GO TO NEXT TEST IF IT IS
2990  022060  104012                        ERROR   12              ;MMR2 DID NOT TRACK PROPERLY
2991
2992
2993
2994                                .SBTTL  ************ ENTRY POINT 3 --- STARTING ADDRESS 210 ************
2995                                .SBTTL  ********** PAGE ADDRESS AND DESCRIPTOR REGISTER TESTS **********
2996                                ;*
2997                                ;*      THIS GROUP OF TESTS IS USED TO CHECK ALL THE PAR'S AND
2998                                ;*      PDR'S; THEIR ADDRESS DECODING ON DIRECT REFERENCES, THEIR
2999                                ;*      READ/WRITE BITS, AND DUAL ADDRESSING WITHIN A GROUP OR BETWEEN
3000                                ;*      TWO GROUPS.
3001                                ;*
3002
3003
3004                                .SBTTL          TEST THAT ALL P.A.R.'S & P.D.R.'S RESPOND
3005                                ;*      THESE NEXT FOUR (4) TESTS VERIFY THAT THERE ARE 4 GROUPS OF
3006                                ;*      8 CONTIGUOUS ADDRESSES IN THE I/O PAGE THAT WILL RESPOND
3007                                ;*      WITHOUT TIMING OUT.  AT THIS POINT THE TEST IS NOT CONCERNED
3008                                ;*      WITH EXACTLY WHAT IS RESPONDING, JUST THAT SOMETHING RESPONDS.
3009                                ;*
3010                                ;**************************************************************
3011                                ;*TEST 10       READ ALL KERNEL PAGE ADDRESS REGISTERS, CHECK FOR TIMEOUT
3012                                ;*
3013                                ;*      THIS TEST DOES A READ FROM ALL THE KERNEL PAGE ADDRESS REGISTERS
3014                                ;*      'ERRVEC' IS SET TO 'TIMEOUT' AT THE BEGINNING OF THE TEST AND
3015                                ;*      IF ANY OF THE 8 KERNEL ADDRESS REGISTERS TIME OUT THEIR I/O
3016                                ;*      PAGE ADDRESS IS REPORTED AND LOGGED.  AT THE END OF THE TEST A
3017                                ;*      SUMMARY OF THE ERRORS IS GIVEN AND 'ERRVEC' IS RE-SET TO 'CPUER'.
3018                                ;*
3019                                ;*      ALL ERRORS IN THIS TEST ARE REPORTED BY SUBROUTINE "TIMEOUT"
3020                                ;*
3021                                ;**************************************************************
3022  022062                        TST10:
3023  022062  000004                       SCOPE
3024  022064  012737  022206  001446       MOV     #TST11,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
3025                                                                ;TEST FOR ESCAPE ON PARITY ERRORS
3026  022072                        ENTPT3:
3027  022072  012737  022114  001110       MOV     #20$,$LPERR     ;SET LOOP ON ERROR POINTER TO 20$
3028  022100  012737  000010  001102       MOV     #10,$TSTNM      ;LOAD TEST NUMBER INTO MEMORY
3029  022106  013777  001102  157026       MOV     $TSTNM,@DISPLAY ;DISPLAY TEST NUMBER FOR THIS TEST
3030  022114  004737  044202          20$: JSR     PC,CLEANUP      ;INITIALIZE ERROR LOCATIONS
3031  022120  012737  044240  000004       MOV     #TIMEOUT,ERRVEC ;SET CPU TRAP VECTOR TO TIMEOUT ROUTINE
3032  022126  012737  022140  001110       MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
3033  022134  012700  172340               MOV     #KIPAR0,R0      ;PUT ADDRESS OF FIRST PAR IN R0
3034  022140  011001                   1$: MOV     (R0),R1         ;THIS WILL TIMEOUT IF REGISTER DECODING BAD
3035  022142  062700  000002               ADD     #2,R0           ;POINT TO NEXT REGISTER
3036  022146  022700  172356               CMP     #KIPAR7,R0      ;SEE IF KIPAR7 HAS BEEN TRIED
```

```
3037  022152  103372                        BHIS    1$                ;BRANCH IF NOT DONE
3038  022154  012737  022114  001110        MOV     #20$,$LPERR       ;PUT LOOP ON ERROR POINTER TO START OF TEST
3039  022162  012737  015162  000004        MOV     #CPUER,ERRVEC     ;RE-SET NORMAL CPU TRAP SERVICE ROUTINE
3040  022170  005737  001432                TST     ERRCNT            ;SEE IF ANY ERRORS OCCURRED ON THIS TEST
3041  022174  001404                        BEQ     TST11             ;;BRANCH TO NEXT TEST IF NO ERRORS
3042  022176  013737  001432  001210        MOV     ERRCNT,$TMP5      ;SAVE NUMBER OF ERRORS FOR TYPEOUT
3043  022204  104013                        ERROR   13                ;SUMMARY OF PAR ERRORS
3044
3045
3046                                  ;;*****************************************************************
3047                                  ;*TEST 11       READ ALL USER PAGE ADDRESS REGISTERS, CHECK FOR TIMEOUT
3048                                  ;;
3049                                  ;;      THIS TEST DOES A READ FROM ALL THE USER PAGE ADDRESS
3050                                  ;;      REGISTERS.  'ERRVEC' IS SET TO 'TIMEOUT' AT THE BEGINNING OF THE
3051                                  ;;      TEST, AND IF ANY OF THE 8 USER ADDRESS REGISTERS TIME OUT
3052                                  ;;      THEIR I/O PAGE ADDRESS IS REPORTED AND LOGGED.  AT THE END
3053                                  ;;      OF THE TEST A SUMMARY OF ERRORS IS GIVEN AND 'ERRVEC' IS SET TO
3054                                  ;;      'CPUER'.
3055                                  ;;      ALL ERRORS IN THIS TEST ARE REPORTED BY SUBROUTINE "TIMEOUT"
3056                                  ;;
3057                                  ;;*****************************************************************
3058  022206                          TST11:
3059  022206  000004                        SCOPE
3060  022210  012737  022310  001446        MOV     #TST12,NXTTST     ;SAVE STARTING ADDRESS OF NEXT
3061                                                                   ;TEST FOR ESCAPE ON PARITY ERRORS
3062  022216  004737  044202          20$:  JSR     PC,CLEANUP        ;INITIALIZE ERROR LOCATIONS
3063  022222  012737  044240  000004        MOV     #TIMEOUT,ERRVEC   ;SET CPU TRAP VECTOR TO TIMEOUT ROUTINE
3064  022230  012737  022242  001110        MOV     #1$,$LPERR        ;SET LOOP ON ERROR POINTER TO 1$
3065  022236  012700  177640                MOV     #UIPAR0,R0        ;PUT ADDRESS OF FIRST PAR IN R0
3066  022242  011001                   1$:  MOV     (R0),R1           ;THIS WILL TIMEOUT IF REGISTER DECODING BAD
3067  022244  062700  000002                ADD     #2,R0             ;POINT TO NEXT REGISTER
3068  022250  022700  177656                CMP     #UIPAR7,R0        ;SEE IF UIPAR7 HAS BEEN TRIED
3069  022254  103372                        BHIS    1$                ;BRANCH IF NOT DONE
3070  022256  012737  022216  001110        MOV     #20$,$LPERR       ;PUT LOOP ON ERROR POINTER TO START OF TEST
3071  022264  012737  015162  000004        MOV     #CPUER,ERRVEC     ;RE-SET NORMAL CPU TRAP SERVICE ROUTINE
3072  022272  005737  001432                TST     ERRCNT            ;SEE IF ANY ERRORS OCCURRED ON THIS TEST
3073  022276  001404                        BEQ     TST12             ;;BRANCH TO NEXT TEST IF NO ERRORS
3074  022300  013737  001432  001210        MOV     ERRCNT,$TMP5      ;SAVE NUMBER OF ERRORS FOR TYPEOUT
3075  022306  104013                        ERROR   13                ;SUMMARY OF PAR ERRORS
3076
3077                                  ;;*****************************************************************
3078                                  ;*TEST 12       READ ALL KERNEL PAGE DESCRIPTOR REGISTERS, CHECK FOR TIMEOUT
3079                                  ;;
3080                                  ;;      THIS TEST DOES A READ FROM ALL THE KERNEL PAGE DESCRIPTOR
3081                                  ;;      REGISTERS.  'ERRVEC' IS SET TO 'TIMEOUT' AT THE BEGINNING OF THE
3082                                  ;;      TEST, AND IF ANY OF THE 8 KERNEL DESCRIPTOR REGISTERS TIME OUT
3083                                  ;;      THEIR I/O PAGE ADDRESS IS REPORTED AND LOGGED.  AT THE END
3084                                  ;;      OF THE TEST A SUMMARY OF ERRORS IS GIVEN AND 'ERRVEC' IS SET TO
3085                                  ;;      'CPUER'.
3086                                  ;;      ALL ERRORS IN THIS TEST ARE REPORTED BY SUBROUTINE "TIMEOUT"
3087                                  ;;
3088                                  ;;*****************************************************************
3089  022310                          TST12:
3090  022310  000004                        SCOPE
3091  022312  012737  022412  001446        MOV     #TST13,NXTTST     ;SAVE STARTING ADDRESS OF NEXT
3092                                                                   ;TEST FOR ESCAPE ON PARITY ERRORS
```

```
MAINDEC-11-DQKTA-A  PDP 11/6X MEM. MGMT. DIAG.  MACY11 27(1006)  07-FEB-77  10:37  PAGE 61
DQKTAA.P11    07-FEB-77 10:30            T12    READ ALL KERNEL PAGE DESCRIPTOR REGISTERS, CHECK FOR TIMEOUT

  3093  022320  004737  044202          20$:    JSR     PC,CLEANUP      ;INITIALIZE ERROR LOCATIONS
  3094  022324  012737  044240  000004          MOV     #TIMEOUT,ERRVEC ;SET CPU TRAP VECTOR TO TIMEOUT ROUTINE
  3095  022332  012737  022344  001110          MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
  3096  022340  012700  172300                  MOV     #KIPDR0,R0      ;PUT ADDRESS OF FIRST PDR IN R0
  3097  022344  011001                  1$:     MOV     (R0),R1         ;THIS WILL TIMEOUT IF REGISTER DECODING BAD
  3098  022346  062700  000002                  ADD     #2,R0           ;POINT TO NEXT REGISTER
  3099  022352  022700  172316                  CMP     #KIPDR7,R0      ;SEE IF KIPDR7 HAS BEEN TRIED
  3100  022356  103372                          BHIS    1$              ;BRANCH IF NOT DONE
  3101  022360  012737  022320  001110          MOV     #20$,$LPERR     ;PUT LOOP ON ERROR POINTER TO START OF TEST
  3102  022366  012737  015162  000004          MOV     #CPUER,ERRVEC   ;RE-SET NORMAL CPU TRAP SERVICE ROUTINE
  3103  022374  005737  001432                  TST     ERRCNT          ;SEE IF ANY ERRORS OCCURRED ON THIS TEST
  3104  022400  001404                          BEQ     TST13           ;;BRANCH TO NEXT TEST IF NO ERRORS
  3105  022402  013737  001432  001210          MOV     ERRCNT,$TMP5    ;SAVE NUMBER OF ERRORS FOR TYPEOUT
  3106  022410  104013                          ERROR   13              ;SUMMARY OF PDR ERRORS
  3107
  3108
  3109                                  ;;**************************************************************
  3110                                  ;*TEST 13       READ ALL USER PAGE DESCRIPTOR REGISTERS, CHECK FOR TIMEOUT
  3111                                  ;;
  3112                                  ;;      THIS TEST DOES A READ FROM ALL THE USER PAGE DESCRIPTOR
  3113                                  ;;      REGISTERS.  'ERRVEC' IS SET TO 'TIMEOUT' AT THE BEGINNING OF THE
  3114                                  ;;      TEST, AND IF ANY OF THE 8 USER DESCRIPTOR REGISTERS TIME OUT
  3115                                  ;;      THEIR I/O PAGE ADDRESS IS REPORTED AND LOGGED.  AT THE END
  3116                                  ;;      OF THE TEST A SUMMARY OF ERRORS IS GIVEN AND 'ERRVEC' IS SET TO
  3117                                  ;;      'CPUER'.
  3118                                  ;;      ALL ERRORS IN THIS TEST ARE REPORTED BY SUBROUTINE "TIMEOUT"
  3119                                  ;;
  3120                                  ;;**************************************************************
  3121  022412                          TST13:
  3122  022412  000004                          SCOPE
  3123  022414  012737  022514  001446          MOV     #TST14,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
  3124                                                                  ;TEST FOR ESCAPE ON PARITY ERRORS
  3125  022422  004737  044202          20$:    JSR     PC,CLEANUP      ;INITIALIZE ERROR LOCATIONS
  3126  022426  012737  044240  000004          MOV     #TIMEOUT,ERRVEC ;SET CPU TRAP VECTOR TO TIMEOUT ROUTINE
  3127  022434  012737  022446  001110          MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
  3128  022442  012700  177600                  MOV     #UIPDR0,R0      ;PUT ADDRESS OF FIRST PDR IN R0
  3129  022446  011001                  1$:     MOV     (R0),R1         ;THIS WILL TIMEOUT IF REGISTER DECODING BAD
  3130  022450  062700  000002                  ADD     #2,R0           ;POINT TO NEXT REGISTER
  3131  022454  022700  177616                  CMP     #UIPDR7,R0      ;SEE IF UIPDR7 HAS BEEN TRIED
  3132  022460  103372                          BHIS    1$              ;BRANCH IF NOT DONE
  3133  022462  012737  022422  001110          MOV     #20$,$LPERR     ;PUT LOOP ON ERROR POINTER TO START OF TEST
  3134  022470  012737  015162  000004          MOV     #CPUER,ERRVEC   ;RE-SET NORMAL CPU TRAP SERVICE ROUTINE
  3135  022476  005737  001432                  TST     ERRCNT          ;SEE IF ANY ERRORS OCCURRED ON THIS TEST
  3136  022502  001404                          BEQ     TST14           ;;BRANCH TO NEXT TEST IF NO ERRORS
  3137  022504  013737  001432  001210          MOV     ERRCNT,$TMP5    ;SAVE NUMBER OF ERRORS FOR TYPEOUT
  3138  022512  104013                          ERROR   13              ;SUMMARY OF PDR ERRORS
  3139
  3140                                  .SBTTL          TEST FOR DUAL ADDRESSING ON LOAD WITHIN GROUPS
  3141                                  ;*      THESE NEXT FOUR (4) TESTS WILL CHECK FOR DUAL ADDRESSING WITHIN A
  3142                                  ;*      GROUP OF PAR'S OR PDR'S.  FIRST ALL OF THE REGISTERS IN A GROUP
  3143                                  ;*      ARE CLEARED AND READ TO SEE THAT THEY CAN EACH HOLD ZEROES, AND
```

```
                                ;*      THAT THE DATA PATH DOES NOT HAVE A BIT STUCK AT ONE.
                                ;*      THEN ONE REGISTER AT A TIME, WITHIN THAT GROUP, IS LOADED
                                ;*      WITH A NEGATIVE ONE WHILE ALL REGISTERS ARE READ TO SEE
                                ;*      THAT ONLY THE REGISTER UNDER TEST WAS NOT ZERO.
                                ;*
                                ;;*****************************************************************
                                ;*TEST 14       DUAL ADDRESS KERNEL PAGE ADDRESS REGISTERS, ON LOADING
                                ;*
                                ;*      THIS TEST FIRST CLEARS ALL THE KERNEL PAGE ADDRESS REGISTERS,
                                ;*      AND CHECKS TO SEE THAT THEY EACH HOLD ZERO.  THEN, STARTING
                                ;*      WITH ADDRESS REGISTER ZERO, ONE REGISTER AT A TIME
                                ;*      IS LOADED WITH A NEGATIVE ONE.  ALL KERNEL ADDRESS REGISTERS
                                ;*      ARE NOW READ TO SEE THAT ONLY THE REGISTER UNDER TEST IS NON-
                                ;*      ZERO.  INDIVIDUAL ERRORS ARE REPORTED AND A SUMMARY OF THEM
                                ;*      IS GIVEN AT THE END OF THIS TEST.
                                ;*
                                ;*      ALL ERRORS IN THIS TEST ARE REPORTED AND ANALYZED BY SUBROUTINE
                                ;*      "DUALADR".
                                ;;*****************************************************************
022514                          TST14:
022514  000004                          SCOPE
022516  012737  022720  001446          MOV     #TST15,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
                                                                ;TEST FOR ESCAPE ON PARITY ERRORS
022524  004737  044202          20$:    JSR     PC,CLEANUP      ;INITIALIZE ERROR LOCATIONS
022530  012737  022556  001110          MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
022536  012705  172340                  MOV     #KIPAR0,R5      ;PUT ADDRESS OF FIRST PAR IN R5
022542  004737  044164                  JSR     PC,CLRREG       ;CLEAR 8 REGISTERS POINTED TO BY R5
022546  012700  172340                  MOV     #KIPAR0,R0      ;PUT ADDRESS OF FIRST PAR IN R0
022552  012701  000010                  MOV     #10,R1          ;BRANCH COUNT IS 8 DECIMAL
022556  011002                  1$:     MOV     (R0),R2         ;READ PAR TO R2
022560  001401                          BEQ     2$              ;BRANCH IF PAR IS 0
022562  104014                          ERROR   14              ;PAR NOT ZERO
022564  062700  000002          2$:     ADD     #2,R0           ;POINT TO NEXT REGISTER
022570  077106                          SOB     R1,1$           ;BRANCH BACK TO 1$ 7 TIMES
                                ;;
                                ;;  NOW START DUAL ADDRESSING TEST BY LOADING -1 INTO ONE
                                ;;  REGISTER AND READING THE REST TO SEE ONLY THAT REGISTER
                                ;;  IS NON-ZERO. (DROPPED BITS  WILL BE FOUND IN NEXT TEST.)
                                ;;
022572  012737  022604  001110          MOV     #3$,$LPERR      ;SET LOOP ON ERROR POINTER TO 3$
022600  012700  172340                  MOV     #KIPAR0,R0      ;PUT ADDRESS OF FIRST PAR IS R0
022604  012705  172340          3$:     MOV     #KIPAR0,R5      ;LOAD STARTING ADDRESS INTO R5
022610  004737  044164                  JSR     PC,CLRREG       ;CLEAR 8 REGISTERS POINTED TO BY R5
022614  012701  172356                  MOV     #KIPAR7,R1      ;PUT KIPAR7 ADDRESS IN R1
022620  012710  177777                  MOV     #-1,(R0)        ;LOAD REGISTER UNDER TEST
022624  005037  001176          4$:     CLR     $TMP0           ;FLAG TO INDICATE THERE WAS A MATCH
022630  011102                          MOV     (R1),R2         ;READ ALL REGISTERS
022632  001406                          BEQ     6$              ;BRANCH IF REGISTER IS 0
022634  020001                          CMP     R0,R1           ;IS ADDRESS OF NON-ZERO REGISTER SAME
                                                                ;AS THAT OF REGISTER UNDER TEST
022636  001402                          BEQ     5$              ;BRANCH IF ADDRESSES MATCH
022640  004737  044334                  JSR     PC,DUALADR      ;LOG AND REPORT ERRORS
022644  005237  001176          5$:     INC     $TMP0           ;SET FLAG WHEN ADDRESSES MATCH
022650  162701  000002          6$:     SUB     #2,R1           ;POINT TO NEXT REGISTER
022654  022701  172340                  CMP     #KIPAR0,R1      ;SEE IF ALL REGISTERS HAVE BEEN READ
022660  101761                          BLOS    4$              ;BRANCH IF MORE TO READ
```

```
3200  022662  062700  000002          ADD     #2,R0           ;NOW LOAD NEXT REGISTER
3201  022666  022700  172356          CMP     #KIPAR7,R0      ;SEE IF THERE ARE MORE REGISTERS TO TEST
3202  022672  103344                  BHIS    3$              ;BRANCH IF MORE REGISTERS TO TEST
3203  022674  012737  022524  001110  MOV     #20$,$LPERR     ;SET LOOP ON ERROR POINTER TO START OF TEST
3204  022702  005737  001432          TST     ERRCNT          ;SEE IF THERE WERE ANY ERRORS
3205  022706  001404                  BEQ     TST15           ;;BRANCH TO NEXT TEST IF NO ERRORS
3206  022710  013737  001432  001210  MOV     ERRCNT,$TMP5    ;SAVE NUMBER OF ERRORS FOR PAR OUT
3207  022716  104015                  ERROR   15              ;SUMMARY OF DUAL ADDRESSING ERRORS
3208
3209
3210                                  ;;*****************************************************************
3211                                  ;*TEST 15       DUAL ADDRESS USER PAGE ADDRESS REGISTERS, ON LOADING
3212                                  ;;
3213                                  ;;      THIS TEST FIRST CLEARS ALL THE USER PAGE ADDRESS REGISTERS
3214                                  ;;      AND CHECKS TO SEE THAT THEY EACH HOLD ZERO.  THEN, STARTING WITH
3215                                  ;;      ADDRESS REGISTER ZERO, ONE REGISTER AT A TIME IS
3216                                  ;;      LOADED WITH A NEGATIVE ONE.  ALL USER ADDRESS REGISTERS
3217                                  ;;      ARE NOW READ TO SEE THAT ONLY THE ONE UNDER TEST IS NON-ZERO.
3218                                  ;;      INDIVIDUAL ERRORS ARE REPORTED AND A SUMMARY OF THEM IS GIVEN AT
3219                                  ;;      THE END OF THIS TEST.
3220                                  ;;
3221                                  ;;      ALL ERRORS IN THIS TEST ARE REPORTED AND ANALYZED BY SUBROUTINE
3222                                  ;;      "DUALADR".
3223                                  ;;*****************************************************************
3224  022720                          TST15:
3225  022720  000004                          SCOPE
3226  022722  012737  023124  001446          MOV     #TST16,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
3227                                                                  ;TEST FOR ESCAPE ON PARITY ERRORS
3228  022730  004737  044202          20$:    JSR     PC,CLEANUP      ;INITIALIZE ERROR LOCATIONS
3229  022734  012737  022762  001110          MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
3230  022742  012705  177640                  MOV     #UIPAR0,R5      ;PUT ADDRESS OF FIRST PAR IN R5
3231  022746  004737  044164                  JSR     PC,CLRREG       ;CLEAR 8 REGISTERS POINTED TO BY R5
3232  022752  012700  177640                  MOV     #UIPAR0,R0      ;PUT ADDRESS OF FIRST PAR IN R0
3233  022756  012701  000010                  MOV     #10,R1          ;BRANCH COUNT IS 8 DECIMAL
3234  022762  011002                  1$:     MOV     (R0),R2         ;READ PAR TO R2
3235  022764  001401                          BEQ     2$              ;BRANCH IF PAR IS 0
3236  022766  104014                          ERROR   14              ;PAR NOT ZERO
3237  022770  062700  000002          2$:     ADD     #2,R0           ;POINT TO NEXT REGISTER
3238  022774  077106                          SOB     R1,1$           ;BRANCH BACK TO 1$ 7 TIMES
3239                                  ;;
3240                                  ;;  NOW START DUAL ADDRESSING TEST BY LOADING -1 INTO ONE
3241                                  ;;  REGISTER AND READING THE REST TO SEE ONLY THAT REGISTER
3242                                  ;;  IS NON-ZERO. (DROPPED BITS  WILL BE FOUND IN NEXT TEST.)
3243                                  ;;
3244  022776  012737  023010  001110          MOV     #3$,$LPERR      ;SET LOOP ON ERROR POINTER TO 3$
3245  023004  012700  177640                  MOV     #UIPAR0,R0      ;PUT ADDRESS OF FIRST PAR IS R0
3246  023010  012705  177640          3$:     MOV     #UIPAR0,R5      ;LOAD STARTING ADDRESS INTO R5
3247  023014  004737  044164                  JSR     PC,CLRREG       ;CLEAR 8 REGISTERS POINTED TO BY R5
3248  023020  012701  177656                  MOV     #UIPAR7,R1      ;PUT UIPAR7 ADDRESS IN R1
3249  023024  012710  177777                  MOV     #-1,(R0)        ;LOAD REGISTER UNDER TEST
3250  023030  005037  001176          4$:     CLR     $TMP0           ;FLAG TO INDICATE THERE WAS A MATCH
3251  023034  011102                          MOV     (R1),R2         ;READ ALL REGISTERS
3252  023036  001406                          BEQ     6$              ;BRANCH IF REGISTER IS 0
3253  023040  020001                          CMP     R0,R1           ;IS ADDRESS OF NON-ZERO REGISTER SAME
3254                                                                  ;AS THAT OF REGISTER UNDER TEST
3255  023042  001402                          BEQ     5$              ;BRANCH IF ADDRESSES MATCH
```

```
3256  023044  004737  044334          JSR     PC,DUALADR      ;LOG AND REPORT ERRORS
3257  023050  005237  001176  5$:     INC     $TMP0           ;SET FLAG WHEN ADDRESSES MATCH
3258  023054  162701  000002  6$:     SUB     #2,R1           ;POINT TO NEXT REGISTER
3259  023060  022701  177640          CMP     #UIPAR0,R1      ;SEE IF ALL REGISTERS HAVE BEEN READ
3260  023064  101761                  BLOS    4$              ;BRANCH IF MORE TO READ
3261  023066  062700  000002          ADD     #2,R0           ;NOW LOAD NEXT REGISTER
3262  023072  022700  177656          CMP     #UIPAR7,R0      ;SEE IF THERE ARE MORE REGISTERS TO TEST
3263  023076  103344                  BHIS    3$              ;BRANCH IF MORE REGISTERS TO TEST
3264  023100  012737  022730  001110  MOV     #20$,$LPERR     ;SET LOOP ON ERROR POINTER TO START OF TEST
3265  023106  005737  001432          TST     ERRCNT          ;SEE IF THERE WERE ANY ERRORS
3266  023112  001404                  BEQ     TST16           ;;BRANCH TO NEXT TEST IF NO ERRORS
3267  023114  013737  001432  001210  MOV     ERRCNT,$TMP5    ;SAVE NUMBER OF ERRORS FOR PAR OUT
3268  023122  104015                  ERROR   15              ;SUMMARY OF DUAL ADDRESSING ERRORS
3269
3270                                  ;;**************************************************************
3271                                  ;;TEST 16       DUAL ADDRESS KERNEL PAGE DESCRIPTOR REGISTERS, ON LOADING
3272                                  ;;
3273                                  ;;      THIS TEST FIRST CLEARS ALL THE KERNEL PAGE DESCRIPTOR REGISTERS
3274                                  ;;      AND CHECKS TO SEE THAT THEY EACH HOLD ZERO.  THEN, STARTING WITH
3275                                  ;;      DESCRIPTOR REGISTER ZERO, ONE REGISTER AT A TIME IS
3276                                  ;;      LOADED WITH A NEGATIVE ONE.  ALL KERNEL DESCRIPTOR REGISTERS
3277                                  ;;      ARE NOW READ TO SEE THAT ONLY THE ONE UNDER TEST IS NON-ZERO.
3278                                  ;;      INDIVIDUAL ERRORS ARE REPORTED AND A SUMMARY OF THEM IS GIVEN AT
3279                                  ;;      THE END OF THIS TEST.
3280                                  ;;
3281                                  ;;      ALL ERRORS IN THIS TEST ARE REPORTED AND ANALYZED BY SUBROUTINE
3282                                  ;;      "DUALADR".
3283                                  ;;**************************************************************
3284  023124                          TST16:
3285  023124  000004                          SCOPE
3286  023126  012737  023330  001446          MOV     #TST17,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
3287                                                                  ;TEST FOR ESCAPE ON PARITY ERRORS
3288  023134  004737  044202  20$:     JSR     PC,CLEANUP      ;INITIALIZE ERROR LOCATIONS
3289  023140  012737  023166  001110          MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
3290  023146  012705  172300                  MOV     #KIPDR0,R5      ;PUT ADDRESS OF FIRST PDR IN R5
3291  023152  004737  044164                  JSR     PC,CLRREG       ;CLEAR 8 REGISTERS POINTED TO BY R5
3292  023156  012700  172300                  MOV     #KIPDR0,R0      ;PUT ADDRESS OF FIRST PDR IN R0
3293  023162  012701  000010                  MOV     #10,R1          ;BRANCH COUNT IS 8 DECIMAL
3294  023166  011002                  1$:     MOV     (R0),R2         ;READ PDR TO R2
3295  023170  001401                          BEQ     2$              ;BRANCH IF PDR IS 0
3296  023172  104014                          ERROR   14              ;PDR NOT ZERO
3297  023174  062700  000002          2$:     ADD     #2,R0           ;POINT TO NEXT REGISTER
3298  023200  077106                          SOB     R1,1$           ;BRANCH BACK TO 1$ 7 TIMES
3299                                  ;;
3300                                  ;;  NOW START DUAL ADDRESSING TEST BY LOADING -1 INTO ONE
3301                                  ;;  REGISTER AND READING THE REST TO SEE ONLY THAT REGISTER
3302                                  ;;  IS NON-ZERO. (DROPPED BITS  WILL BE FOUND IN NEXT TEST.)
3303                                  ;;
3304  023202  012737  023214  001110          MOV     #3$,$LPERR      ;SET LOOP ON ERROR POINTER TO 3$
3305  023210  012700  172300                  MOV     #KIPDR0,R0      ;PUT ADDRESS OF FIRST PDR IS R0
3306  023214  012705  172300          3$:     MOV     #KIPDR0,R5      ;LOAD STARTING ADDRESS INTO R5
3307  023220  004737  044164                  JSR     PC,CLRREG       ;CLEAR 8 REGISTERS POINTED TO BY R5
3308  023224  012701  172316                  MOV     #KIPDR7,R1      ;PUT KIPDR7 ADDRESS IN R1
3309  023230  012710  177777                  MOV     #-1,(R0)        ;LOAD REGISTER UNDER TEST
3310  023234  005037  001176          4$:     CLR     $TMP0           ;FLAG TO INDICATE THERE WAS A MATCH
3311  023240  011102                          MOV     (R1),R2         ;READ ALL REGISTERS
```

```
3312  023242  001406                          BEQ     6$              ;BRANCH IF REGISTER IS 0
3313  023244  020001                          CMP     R0,R1           ;IS ADDRESS OF NON-ZERO REGISTER SAME
3314                                                                  ;AS THAT OF REGISTER UNDER TEST
3315  023246  001402                          BEQ     5$              ;BRANCH IF ADDRESSES MATCH
3316  023250  004737  044334                  JSR     PC,DUALADR      ;LOG AND REPORT ERRORS
3317  023254  005237  001176          5$:     INC     $TMP0           ;SET FLAG WHEN ADDRESSES MATCH
3318  023260  162701  000002          6$:     SUB     #2,R1           ;POINT TO NEXT REGISTER
3319  023264  022701  172300                  CMP     #KIPDR0,R1      ;SEE IF ALL REGISTERS HAVE BEEN READ
3320  023270  101761                          BLOS    4$              ;BRANCH IF MORE TO READ
3321  023272  062700  000002                  ADD     #2,R0           ;NOW LOAD NEXT REGISTER
3322  023276  022700  172316                  CMP     #KIPDR7,R0      ;SEE IF THERE ARE MORE REGISTERS TO TEST
3323  023302  103344                          BHIS    3$              ;BRANCH IF MORE REGISTERS TO TEST
3324  023304  012737  023134  001110          MOV     #20$,$LPERR     ;SET LOOP ON ERROR POINTER TO START OF TEST
3325  023312  005737  001432                  TST     ERRCNT          ;SEE IF THERE WERE ANY ERRORS
3326  023316  001404                          BEQ     TST17           ;;BRANCH TO NEXT TEST IF NO ERRORS
3327  023320  013737  001432  001210          MOV     ERRCNT,$TMP5    ;SAVE NUMBER OF ERRORS FOR PDR OUT
3328  023326  104015                          ERROR   15              ;SUMMARY OF DUAL ADDRESSING ERRORS
3329
3330
3331                                  ;;**************************************************************
3332                                  ;*TEST 17       DUAL ADDRESS USER PAGE DESCRIPTOR REGISTERS, ON LOADING
3333                                  ;;
3334                                  ;;      THIS TEST FIRST CLEARS ALL THE USER PAGE DESCRIPTOR REGISTERS
3335                                  ;;      AND CHECKS TO SEE THAT THEY EACH HOLD ZERO.  THEN, STARTING WITH
3336                                  ;;      DESCRIPTOR REGISTER ZERO, ONE REGISTER AT A TIME IS
3337                                  ;;      LOADED WITH A NEGATIVE ONE.  ALL USER DESCRIPTOR REGISTERS
3338                                  ;;      ARE NOW READ TO SEE THAT ONLY THE ONE UNDER TEST IS NON-ZERO.
3339                                  ;;      INDIVIDUAL ERRORS ARE REPORTED AND A SUMMARY OF THEM IS GIVEN AT
3340                                  ;;      THE END OF THIS TEST.
3341                                  ;;
3342                                  ;;      ALL ERRORS IN THIS TEST ARE REPORTED AND ANALYZED BY SUBROUTINE
3343                                  ;;      "DUALADR".
3344                                  ;;**************************************************************
3345  023330                          TST17:
3346  023330  000004                          SCOPE
3347  023332  012737  023534  001446          MOV     #TST20,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
3348                                                                  ;TEST FOR ESCAPE ON PARITY ERRORS
3349  023340  004737  044202          20$:    JSR     PC,CLEANUP      ;INITIALIZE ERROR LOCATIONS
3350  023344  012737  023372  001110          MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
3351  023352  012705  177600                  MOV     #UIPDR0,R5      ;PUT ADDRESS OF FIRST PDR IN R5
3352  023356  004737  044164                  JSR     PC,CLRREG       ;CLEAR 8 REGISTERS POINTED TO BY R5
3353  023362  012700  177600                  MOV     #UIPDR0,R0      ;PUT ADDRESS OF FIRST PDR IN R0
3354  023366  012701  000010                  MOV     #10,R1          ;BRANCH COUNT IS 8 DECIMAL
3355  023372  011002                  1$:     MOV     (R0),R2         ;READ PDR TO R2
3356  023374  001401                          BEQ     2$              ;BRANCH IF PDR IS 0
3357  023376  104014                          ERROR   14              ;PDR NOT ZERO
3358  023400  062700  000002          2$:     ADD     #2,R0           ;POINT TO NEXT REGISTER
3359  023404  077106                          SOB     R1,1$           ;BRANCH BACK TO 1$ 7 TIMES
3360                                  ;;
3361                                  ;;      NOW START DUAL ADDRESSING TEST BY LOADING -1 INTO ONE
3362                                  ;;      REGISTER AND READING THE REST TO SEE ONLY THAT REGISTER
3363                                  ;;      IS NON-ZERO. (DROPPED BITS  WILL BE FOUND IN NEXT TEST.)
3364                                  ;;
3365  023406  012737  023420  001110          MOV     #3$,$LPERR      ;SET LOOP ON ERROR POINTER TO 3$
3366  023414  012700  177600                  MOV     #UIPDR0,R0      ;PUT ADDRESS OF FIRST PDR IS R0
3367  023420  012705  177600          3$:     MOV     #UIPDR0,R5      ;LOAD STARTING ADDRESS INTO R5
```

```
3368  023424  004737  044164                      JSR     PC,CLRREG       ;CLEAR 8 REGISTERS POINTED TO BY R5
3369  023430  012701  177616                      MOV     #UIPDR7,R1      ;PUT UIPDR7 ADDRESS IN R1
3370  023434  012710  177777                      MOV     #-1,(R0)        ;LOAD REGISTER UNDER TEST
3371  023440  005037  001176            4$:     CLR     $TMP0           ;FLAG TO INDICATE THERE WAS A MATCH
3372  023444  011102                              MOV     (R1),R2         ;READ ALL REGISTERS
3373  023446  001406                              BEQ     6$              ;BRANCH IF REGISTER IS 0
3374  023450  020001                              CMP     R0,R1           ;IS ADDRESS OF NON-ZERO REGISTER SAME
3375                                                                        ;AS THAT OF REGISTER UNDER TEST
3376  023452  001402                              BEQ     5$              ;BRANCH IF ADDRESSES MATCH
3377  023454  004737  044334                      JSR     PC,DUALADR      ;LOG AND REPORT ERRORS
3378  023460  005237  001176            5$:     INC     $TMP0           ;SET FLAG WHEN ADDRESSES MATCH
3379  023464  162701  000002            6$:     SUB     #2,R1           ;POINT TO NEXT REGISTER
3380  023470  022701  177600                      CMP     #UIPDR0,R1      ;SEE IF ALL REGISTERS HAVE BEEN READ
3381  023474  101761                              BLOS    4$              ;BRANCH IF MORE TO READ
3382  023476  062700  000002                      ADD     #2,R0           ;NOW LOAD NEXT REGISTER
3383  023502  022700  177616                      CMP     #UIPDR7,R0      ;SEE IF THERE ARE MORE REGISTERS TO TEST
3384  023506  103344                              BHIS    3$              ;BRANCH IF MORE REGISTERS TO TEST
3385  023510  012737  023340  001110              MOV     #20$,$LPERR     ;SET LOOP ON ERROR POINTER TO START OF TEST
3386  023516  005737  001432                      TST     ERRCNT          ;SEE IF THERE WERE ANY ERRORS
3387  023522  001404                              BEQ     TST20           ;;BRANCH TO NEXT TEST IF NO ERRORS
3388  023524  013737  001432  001210              MOV     ERRCNT,$TMP5    ;SAVE NUMBER OF ERRORS FOR PDR OUT
3389  023532  104015                              ERROR   15              ;SUMMARY OF DUAL ADDRESSING ERRORS
3390
3391                                      .SBTTL          TEST FOR BAD READ/WRITE BITS IN P.A.R.'S & P.D.R.'S
3392                                      ;*      THESE NEXT FOUR (4) TESTS CHECK FOR BAD BITS IN THE MEMORY CHIPS
3393                                      ;*      THAT MAKE UP THE PAR'S AND PDR'S.  THE REGISTERS ARE LOADED WITH
3394                                      ;*      ZERO AND MODIFIED BY "401" UNTIL 177777 IS REACHED IN EACH ONE
3395                                      ;*      (THE NUMBER 401 WAS CHOSEN FOR FASTER RUN-TIME).  THE BITS THAT
3396                                      ;*      ARE NOT IMPLEMENTED OR THAT ARE NOT READ/WRITE ARE MASKED OUT
3397                                      ;*      OF THE DATA COMPARE.  A LOG OF MULTIPLE ERRORS IS KEPT FOR EACH
3398                                      ;*      GROUP AND IT IS REPORTED AT THE CONCLUSION OF EACH TEST.
3399                                      ;*
3400                                      ;**************************************************************
3401                                      ;*TEST 20       COUNT PATTERN IN KERNEL PAGE ADDRESS REGISTERS
3402                                      ;*
3403                                      ;*      THIS TEST RUNS A COUNT PATTERN THRU THE KERNEL PAGE ADDRESS
3404                                      ;*      REGISTERS.  SINCE BITS <15:12> ARE NOT IMPLEMENTED THEY ARE
3405                                      ;*      MASKED OUT OF THE DATA COMPARE.    THESE BITS ARE STILL SENT
3406                                      ;*      TO THE ADDRESS REGISTER TO FIND BAD ETCHES.  IF THE COUNT PATTERN
3407                                      ;*      DOES NOT MATCH THE DATA RECEVIED, THE REGISTER ADDRESS, DATA
3408                                      ;*      PATTERN, AND BAD DATA ARE REPORTED.  AT THE END OF THE TEST A
3409                                      ;*      SUMMARY OF THE ERRORS IS GIVEN.
3410                                      ;*
3411                                      ;*      ALL ERRORS FOUND BY THIS TEST ARE REPORTED AND ANALYZED BY
3412                                      ;*      SUBROUTINE "PARCOUNT".
3413                                      ;*
3414                                      ;**************************************************************
3415  023534                              TST20:
3416  023534  000004                              SCOPE
3417  023536  012737  023666  001446              MOV     #TST21,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
3418                                                                        ;TEST FOR ESCAPE ON PARITY ERRORS
3419  023544  012737  000012  001212              MOV     #12,$TIMES      ;;DO 12 ITERATIONS
3420  023552  004737  044202            20$:    JSR     PC,CLEANUP      ;INITIALIZE ERROR LOCATIONS
3421  023556  012737  023572  001110              MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
3422  023564  005001                              CLR     R1              ;CLEAR REGISTER TO HOLD COUNT PATTERN
3423  023566  012700  172340            1$:     MOV     #KIPAR0,R0      ;PUT ADDRESS OF FIRST REGISTER IS R0
```

```
3424  023572  010110                    2$:     MOV     R1,(R0)         ;LOAD COUNT INTO REGISTER
3425  023574  011002                            MOV     (R0),R2         ;READ REGISTER BACK TO R2
3426  023576  010104                            MOV     R1,R4           ;PUT PATTERN IS R4
3427  023600  042704  170000                    BIC     #170000,R4      ;CLEAR BITS NOT FOUND IN REGISTER.
3428  023604  020402                            CMP     R4,R2           ;SEE IF DATA MATCHES PATTERN
3429  023606  001402                            BEQ     3$              ;BRANCH IF DATA IS GOOD
3430  023610  004737  044410                    JSR     PC,PARCOUNT     ;LOG AND REPORT COUNT ERROR
3431  023614  062700  000002            3$:     ADD     #2,R0           ;POINT TO  NEXT REGISTER
3432  023620  022700  172356                    CMP     #KIPAR7,R0      ;SEE IF YOU PASSED THE KIPAR7 PAR
3433  023624  103362                            BHIS    2$              ;BRANCH IF MORE PAR''S TO TEST
3434  023626  022701  177777                    CMP     #177777,R1      ;SEE IF COUNT HAS REACHED 177777
3435  023632  001403                            BEQ     4$              ;BRANCH IF COUNT IS 177777
3436  023634  062701  000401                    ADD     #401,R1         ;INCREASE COUNT PATTERN
3437  023640  000752                            BR      1$              ;BRANCH TO CONTINUE TEST
3438  023642  012737  023552  001110    4$:     MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
3439  023650  005737  001432                    TST     ERRCNT          ;SEE IF THERE WERE ANY ERRORS
3440  023654  001404                            BEQ     TST21           ;;BRANCH TO NEXT TEST IF NO ERRORS
3441  023656  013737  001432  001210            MOV     ERRCNT,$TMP5    ;SAVE NUMBER OF ERRORS FOR TYPEOUT
3442  023664  104016                            ERROR   16              ;SUMMARY OF COUNT PATTERN FAILURES
3443
3444
3445                                    ;;*****************************************************************
3446                                    ;*TEST 21       COUNT PATTERN IN USER PAGE ADDRESS REGISTERS
3447                                    ;;
3448                                    ;;      THIS TEST RUNS A COUNT PATTERN THRU THE USER PAGE ADDRESS
3449                                    ;;      REGISTERS.  SINCE BITS <15:12> ARE NOT IMPLEMENTED THEY ARE
3450                                    ;*      MASKED OUT OF THE DATA COMPARE.  THESE BITS ARE STILL SENT
3451                                    ;*      TO THE ADDRESS REGISTER TO FIND BAD ETCHES.  IF THE COUNT PATTERN
3452                                    ;;      DOES NOT MATCH THE DATA RECEVIED, THE REGISTER ADDRESS, DATA
3453                                    ;;      PATTERN, AND BAD DATA ARE REPORTED.  AT THE END OF THE TEST A
3454                                    ;;      SUMMARY OF THE ERRORS IS GIVEN.
3455                                    ;;
3456                                    ;;      ALL ERRORS FOUND BY THIS TEST ARE REPORTED AND ANALYZED BY
3457                                    ;;      SUBROUTINE "PARCOUNT".
3458                                    ;;
3459                                    ;;*****************************************************************
3460  023666                            TST21:
3461  023666  000004                            SCOPE
3462  023670  012737  024020  001446            MOV     #TST22,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
3463                                                                    ;TEST FOR ESCAPE ON PARITY ERRORS
3464  023676  012737  000012  001212            MOV     #12,$TIMES      ;;DO 12 ITERATIONS
3465  023704  004737  044202            20$:    JSR     PC,CLEANUP      ;INITIALIZE ERROR LOCATIONS
3466  023710  012737  023724  001110            MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
3467  023716  005001                            CLR     R1              ;CLEAR REGISTER TO HOLD COUNT PATTERN
3468  023720  012700  177640            1$:     MOV     #UIPAR0,R0      ;PUT ADDRESS OF FIRST REGISTER IS R0
3469  023724  010110                    2$:     MOV     R1,(R0)         ;LOAD COUNT INTO REGISTER
3470  023726  011002                            MOV     (R0),R2         ;READ REGISTER BACK TO R2
3471  023730  010104                            MOV     R1,R4           ;PUT PATTERN IS R4
3472  023732  042704  170000                    BIC     #170000,R4      ;CLEAR BITS NOT FOUND IN REGISTER.
3473  023736  020402                            CMP     R4,R2           ;SEE IF DATA MATCHES PATTERN
3474  023740  001402                            BEQ     3$              ;BRANCH IF DATA IS GOOD
3475  023742  004737  044410                    JSR     PC,PARCOUNT     ;LOG AND REPORT COUNT ERROR
3476  023746  062700  000002            3$:     ADD     #2,R0           ;POINT TO  NEXT REGISTER
3477  023752  022700  177656                    CMP     #UIPAR7,R0      ;SEE IF YOU PASSED THE UIPAR7 PAR
3478  023756  103362                            BHIS    2$              ;BRANCH IF MORE PAR''S TO TEST
3479  023760  022701  177777                    CMP     #177777,R1      ;SEE IF COUNT HAS REACHED 177777
```

```
3480  023764  001403                        BEQ     4$              ;BRANCH IF COUNT IS 177777
3481  023766  062701  000401                ADD     #401,R1         ;INCREASE COUNT PATTERN
3482  023772  000752                        BR      1$              ;BRANCH TO CONTINUE TEST
3483  023774  012737  023704  001110  4$:   MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
3484  024002  005737  001432                TST     ERRCNT          ;SEE IF THERE WERE ANY ERRORS
3485  024006  001404                        BEQ     TST22           ;;BRANCH TO NEXT TEST IF NO ERRORS
3486  024010  013737  001432  001210        MOV     ERRCNT,$TMP5    ;SAVE NUMBER OF ERRORS FOR TYPEOUT
3487  024016  104016                        ERROR   16              ;SUMMARY OF COUNT PATTERN FAILURES
3488
3489                                ;;**********************************************************************
3490                                ;*TEST 22       COUNT PATTERN IN KERNEL PAGE DESCRIPTOR REGISTERS
3491                                ;*
3492                                ;*      THIS TEST RUNS A COUNT PATTERN THRU THE KERNEL PAGE DESCRIPTOR
3493                                ;*      REGISTERS.  SINCE BITS <15>,<07>,<05:04> AND <00> ARE NOT IMPLEMENTED
3494                                ;*      AND BITS <06> CANNOT BE SET BY DIRECT LOAD, THEY ARE MASKED
3495                                ;*      OUT OF THE DATA COMPARE.  THESE BITS ARE STILL SENT TO THE
3496                                ;*      DESCRIPTOR REGISTER TO FIND BAD ETCHES.  IF THE COUNT PATTERN
3497                                ;*      DOES NOT MATCH THE DATA RECEVIED, THE REGISTER ADDRESS, DATA
3498                                ;*      PATTERN, AND BAD DATA ARE REPORTED.  AT THE END OF THE TEST A
3499                                ;*      SUMMARY OF THE ERRORS IS GIVEN.
3500                                ;*
3501                                ;;**********************************************************************
3502  024020                        TST22:
3503  024020  000004                        SCOPE
3504  024022  012737  024152  001446        MOV     #TST23,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
3505                                                                ;TEST FOR ESCAPE ON PARITY ERRORS
3506  024030  012737  000012  001212        MOV     #12,$TIMES      ;;DO 12 ITERATIONS
3507  024036  004737  044202          20$:  JSR     PC,CLEANUP      ;INITIALIZE ERROR LOCATIONS
3508  024042  012737  024056  001110        MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
3509  024050  005001                        CLR     R1              ;CLEAR REGISTER TO HOLD COUNT PATTERN
3510  024052  012700  172300          1$:   MOV     #KIPDR0,R0      ;PUT ADDRESS OF FIRST REGISTER IS R0
3511  024056  010110                  2$:   MOV     R1,(R0)         ;LOAD COUNT INTO REGISTER
3512  024060  011002                        MOV     (R0),R2         ;READ REGISTER BACK TO R2
3513  024062  010104                        MOV     R1,R4           ;PUT PATTERN IS R4
3514  024064  042704  100361                BIC     #100361,R4      ;CLEAR BITS NOT FOUND IN REGISTER.
3515  024070  020402                        CMP     R4,R2           ;SEE IF DATA MATCHES PATTERN
3516  024072  001402                        BEQ     3$              ;BRANCH IF DATA IS GOOD
3517  024074  004737  044410                JSR     PC,PARCOUNT     ;LOG AND REPORT COUNT ERROR
3518  024100  062700  000002          3$:   ADD     #2,R0           ;POINT TO  NEXT REGISTER
3519  024104  022700  172316                CMP     #KIPDR7,R0      ;SEE IF YOU PASSED THE KIPDR7 PDR
3520  024110  103362                        BHIS    2$              ;BRANCH IF MORE PDR''S TO TEST
3521  024112  022701  177777                CMP     #177777,R1      ;SEE IF COUNT HAS REACHED 177777
3522  024116  001403                        BEQ     4$              ;BRANCH IF COUNT IS 177777
3523  024120  062701  000401                ADD     #401,R1         ;INCREASE COUNT PATTERN
3524  024124  000752                        BR      1$              ;BRANCH TO CONTINUE TEST
3525  024126  012737  024036  001110  4$:   MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
3526  024134  005737  001432                TST     ERRCNT          ;SEE IF THERE WERE ANY ERRORS
3527  024140  001404                        BEQ     TST23           ;;BRANCH TO NEXT TEST IF NO ERRORS
3528  024142  013737  001432  001210        MOV     ERRCNT,$TMP5    ;SAVE NUMBER OF ERRORS FOR TYPEOUT
3529  024150  104016                        ERROR   16              ;SUMMARY OF COUNT PATTERN FAILURES
3530
3531
3532                                ;;**********************************************************************
3533                                ;*TEST 23       COUNT PATTERN IN USER PAGE DESCRIPTOR REGISTERS
3534                                ;;
3535                                ;;      THIS TEST RUNS A COUNT PATTERN THRU THE USER PAGE DESCRIPTOR
```

```
                                        ;;      REGISTERS.  SINCE BITS (15),(07),(05:04), AND (00) ARE NOT IMPLEMENTED
                                        ;;      AND BITS (06) CANNOT BE SET BY DIRECT LOAD, THEY ARE MASKED
                                        ;;      OUT OF THE DATA COMPARE.  THESE BITS ARE STILL SENT TO THE
                                        ;;      DESCRIPTOR REGISTER TO FIND BAD ETCHES.  IF THE COUNT PATTERN
                                        ;;      DOES NOT MATCH THE DATA RECEVIED, THE REGISTER ADDRESS, DATA
                                        ;;      PATTERN, AND BAD DATA ARE REPORTED.  AT THE END OF THE TEST A
                                        ;;      SUMMARY OF THE ERRORS IS GIVEN.
                                        ;;
                                        ;;**************************************************************************
024152                                  TST23:
024152  000004                          SCOPE
024154  012737  024304  001446          MOV     #TST24,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
                                                                ;TEST FOR ESCAPE ON PARITY ERRORS
024162  012737  000012  001212          MOV     #12,$TIMES      ;;DO 12 ITERATIONS
024170  004737  044202          20$:    JSR     PC,CLEANUP      ;INITIALIZE ERROR LOCATIONS
024174  012737  024210  001110          MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
024202  005001                          CLR     R1              ;CLEAR REGISTER TO HOLD COUNT PATTERN
024204  012700  177600          1$:     MOV     #UIPDR0,R0      ;PUT ADDRESS OF FIRST REGISTER IS R0
024210  010110                  2$:     MOV     R1,(R0)         ;LOAD COUNT INTO REGISTER
024212  011002                          MOV     (R0),R2         ;READ REGISTER BACK TO R2
024214  010104                          MOV     R1,R4           ;PUT PATTERN IS R4
024216  042704  100361                  BIC     #100361,R4      ;CLEAR BITS NOT FOUND IN REGISTER.
024222  020402                          CMP     R4,R2           ;SEE IF DATA MATCHES PATTERN
024224  001402                          BEQ     3$              ;BRANCH IF DATA IS GOOD
024226  004737  044410                  JSR     PC,PARCOUNT     ;LOG AND REPORT COUNT ERROR
024232  062700  000002          3$:     ADD     #2,R0           ;POINT TO  NEXT REGISTER
024236  022700  177616                  CMP     #UIPDR7,R0      ;SEE IF YOU PASSED THE UIPDR7 PDR
024242  103362                          BHIS    2$              ;BRANCH IF MORE PDR''S TO TEST
024244  022701  177777                  CMP     #177777,R1      ;SEE IF COUNT HAS REACHED 177777
024250  001403                          BEQ     4$              ;BRANCH IF COUNT IS 177777
024252  062701  000401                  ADD     #401,R1         ;INCREASE COUNT PATTERN
024256  000752                          BR      1$              ;BRANCH TO CONTINUE TEST
024260  012737  024170  001110  4$:     MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
024266  005737  001432                  TST     ERRCNT          ;SEE IF THERE WERE ANY ERRORS
024272  001404                          BEQ     TST24           ;;BRANCH TO NEXT TEST IF NO ERRORS
024274  013737  001432  001210          MOV     ERRCNT,$TMP5    ;SAVE NUMBER OF ERRORS FOR TYPEOUT
024302  104016                          ERROR   16              ;SUMMARY OF COUNT PATTERN FAILURES

                                .SBTTL          TEST FOR CORRECT BYTE ADDRESSING OF P.A.R.'S & P.D.R.'S
                                ;*      THESE NEXT FOUR (4) TESTS ARE USED TO TEST THE LOGIC THAT
                                ;*      ALLOWS BYTE ADDRESSING OF THE PAR'S AND PDR'S.  IN EACH
                                ;*      CASE REGISTER 0 IS USED, SINCE IT IS REALLY
                                ;*      THE WRITE PULSE TO THE REGISTER SET THAT IS BEING TESTED.
                                ;*      IT IS ASSUMED THAT IF ONE REGISTER OF A GROUP FUNCTIONS
                                ;*      PROPERLY THAT THEY ALL WILL, SINCE ALL REGISTERS HAVE BEEN
                                ;*      BIT TESTED EARLIER.
                                ;*
                                ;;**************************************************************************
                                ;*TEST 24       BYTE ADDRESSING OF KERNEL PAGE ADDRESS REGISTERS
                                ;*
                                ;*      THIS TEST CHECKS THE 'WRITE PAR LOW' AND 'WRITE PAR HIGH'
                                ;*      SIGNAL GENERATION FOR KERNEL PAGE ADDRESS REGISTERS.
                                ;*      BOTH SIGNALS ARE CONTROLLED BY THE WRITE REG. DECODE LOGIC.
                                ;*
                                ;;**************************************************************************
024304                          TST24:
```

```
3592  024304  000004                          SCOPE
3593  024306  012737  024416  001446          MOV     #TST25,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
3594                                                                  ;TEST FOR ESCAPE ON PARITY ERRORS
3595  024314  012737  024332  001110  20$:    MOV     #11$,$LPERR     ;SET LOOP ON ERROR POINTER TO 11$
3596  024322  012702  000016                  MOV     #16,R2          ;PUT EXPECTED DATA IN R2
3597  024326  012700  172340                  MOV     #KIPAR0,R0      ;PUT ADDRESS OF REGISTER IN R0
3598  024332  005037  172340          11$:    CLR     KIPAR0          ;CLEAR THE REGISTER UNDER TEST
3599  024336  112710  172016                  MOVB    #172016,(R0)    ;LOAD LOWER BYTE OF REGISTER
3600  024342  013701  172340                  MOV     KIPAR0,R1       ;READ REGISTER INTO R1
3601  024346  020102                          CMP     R1,R2           ;SEE IF ONLY LOWER BYTE WAS WRITTEN
3602  024350  001401                          BEQ     1$              ;BRANCH IF DATA MATCHES
3603  024352  104017                          ERROR   17              ;DIDN'T LOAD CORRECT BYTE
3604  024354  012737  024372  001110  1$:     MOV     #12$,$LPERR     ;SET LOOP ON ERROR POINTER TO 12$
3605  024362  012700  172341                  MOV     #KIPAR0+1,R0    ;POINT TO UPPER BYTE OF REGISTER
3606  024366  012702  007416                  MOV     #007416,R2      ;LOAD EXPECTED DATA IN R2
3607  024372  112710  000017          12$:    MOVB    #17,(R0)        ;WRITE THE UPPER BYTE OF REGISTER
3608  024376  013701  172340                  MOV     KIPAR0,R1       ;READ REGISTER INTO R1
3609  024402  020102                          CMP     R1,R2           ;SEE IF REGISTER HOLDS CORRECT DATA
3610  024404  001401                          BEQ     2$              ;BRANCH TO EXIT IF DATA RIGHT
3611  024406  104017                          ERROR   17              ;DIDN'T LOAD CORRECT BYTE
3612
3613  024410  012737  024314  001110  2$:     MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
3614
3615
3616
3617
3618                                  ;;*****************************************************************
3619                                  ;*TEST 25       BYTE ADDRESSING OF USER PAGE ADDRESS REGISTERS
3620                                  ;;
3621                                  ;;      THIS TEST CHECKS THE 'WRITE PAR LOW' AND 'WRITE PAR HIGH'
3622                                  ;;      SIGNAL GENERATION FOR USER PAGE ADDRESS REGISTERS.
3623                                  ;;      BOTH SIGNALS ARE CONTROLLED BY THE WRITE REG. DECODE LOGIC.
3624                                  ;;
3625                                  ;;*****************************************************************
3626  024416                          TST25:
3627  024416  000004                          SCOPE
3628  024420  012737  024530  001446          MOV     #TST26,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
3629                                                                  ;TEST FOR ESCAPE ON PARITY ERRORS
3630  024426  012737  024444  001110  20$:    MOV     #11$,$LPERR     ;SET LOOP ON ERROR POINTER TO 11$
3631  024434  012702  000016                  MOV     #16,R2          ;PUT EXPECTED DATA IN R2
3632  024440  012700  177640                  MOV     #UIPAR0,R0      ;PUT ADDRESS OF REGISTER IN R0
3633  024444  005037  177640          11$:    CLR     UIPAR0          ;CLEAR THE REGISTER UNDER TEST
3634  024450  112710  172016                  MOVB    #172016,(R0)    ;LOAD LOWER BYTE OF REGISTER
3635  024454  013701  177640                  MOV     UIPAR0,R1       ;READ REGISTER INTO R1
3636  024460  020102                          CMP     R1,R2           ;SEE IF ONLY LOWER BYTE WAS WRITTEN
3637  024462  001401                          BEQ     1$              ;BRANCH IF DATA MATCHES
3638  024464  104017                          ERROR   17              ;DIDN'T LOAD CORRECT BYTE
3639  024466  012737  024504  001110  1$:     MOV     #12$,$LPERR     ;SET LOOP ON ERROR POINTER TO 12$
3640  024474  012700  177641                  MOV     #UIPAR0+1,R0    ;POINT TO UPPER BYTE OF REGISTER
3641  024500  012702  007416                  MOV     #007416,R2      ;LOAD EXPECTED DATA IN R2
3642  024504  112710  000017          12$:    MOVB    #17,(R0)        ;WRITE THE UPPER BYTE OF REGISTER
3643  024510  013701  177640                  MOV     UIPAR0,R1       ;READ REGISTER INTO R1
3644  024514  020102                          CMP     R1,R2           ;SEE IF REGISTER HOLDS CORRECT DATA
3645  024516  001401                          BEQ     2$              ;BRANCH TO EXIT IF DATA RIGHT
3646  024520  104017                          ERROR   17              ;DIDN'T LOAD CORRECT BYTE
3647
```

```
3648  024522  012737  024426  001110  2$:     MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
3649
3650
3651                                          ;;**************************************************************
3652                                          ;*TEST 26       BYTE ADDRESSING OF KERNEL PAGE DESCRIPTOR REGISTERS
3653                                          ;;
3654                                          ;;      THIS TEST CHECKS THE 'WRITE PDR LOW' AND 'WRITE PDR HIGH'
3655                                          ;;      SIGNAL GENERATION FOR KERNEL PAGE ADDRESS REGISTERS.
3656                                          ;;      BOTH SIGNALS ARE CONTROLLED BY THE WRITE REG. DECODE LOGIC.
3657                                          ;;
3658                                          ;;**************************************************************
3659  024530                                  TST26:
3660  024530  000004                                 SCOPE
3661  024532  012737  024642  001446                  MOV     #TST27,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
3662                                                                          ;TEST FOR ESCAPE ON PARITY ERRORS
3663  024540  012737  024556  001110  20$:     MOV     #11$,$LPERR     ;SET LOOP ON ERROR POINTER TO 11$
3664  024546  012702  000016                  MOV     #16,R2          ;PUT EXPECTED DATA IN R2
3665  024552  012700  172300                  MOV     #KIPDR0,R0      ;PUT ADDRESS OF REGISTER IN R0
3666  024556  005037  172300          11$:    CLR     KIPDR0          ;CLEAR THE REGISTER UNDER TEST
3667  024562  112710  172016                  MOVB    #172016,(R0)    ;LOAD LOWER BYTE OF REGISTER
3668  024566  013701  172300                  MOV     KIPDR0,R1       ;READ REGISTER INTO R1
3669  024572  020102                          CMP     R1,R2           ;SEE IF ONLY LOWER BYTE WAS WRITTEN
3670  024574  001401                          BEQ     1$              ;BRANCH IF DATA MATCHES
3671  024576  104017                          ERROR   17              ;DIDN'T LOAD CORRECT BYTE
3672  024600  012737  024616  001110  1$:     MOV     #12$,$LPERR     ;SET LOOP ON ERROR POINTER TO 12$
3673  024606  012700  172301                  MOV     #KIPDR0+1,R0    ;POINT TO UPPER BYTE OF REGISTER
3674  024612  012702  007416                  MOV     #007416,R2      ;LOAD EXPECTED DATA IN R2
3675  024616  112710  000017          12$:    MOVB    #17,(R0)        ;WRITE THE UPPER BYTE OF REGISTER
3676  024622  013701  172300                  MOV     KIPDR0,R1       ;READ REGISTER INTO R1
3677  024626  020102                          CMP     R1,R2           ;SEE IF REGISTER HOLDS CORRECT DATA
3678  024630  001401                          BEQ     2$              ;BRANCH TO EXIT IF DATA RIGHT
3679  024632  104017                          ERROR   17              ;DIDN'T LOAD CORRECT BYTE
3680
3681  024634  012737  024540  001110  2$:     MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
3682
3683
3684
3685
3686                                          ;;**************************************************************
3687                                          ;*TEST 27       BYTE ADDRESSING OF USER PAGE DESCRIPTOR REGISTERS
3688                                          ;;
3689                                          ;;      THIS TEST CHECKS THE 'WRITE PDR LOW' AND 'WRITE PDR HIGH'
3690                                          ;;      SIGNAL GENERATION FOR USER PAGE ADDRESS REGISTERS.
3691                                          ;;      BOTH SIGNALS ARE CONTROLLED BY THE WRITE REG. DECODE LOGIC.
3692                                          ;;
3693                                          ;;**************************************************************
3694  024642                                  TST27:
3695  024642  000004                                 SCOPE
3696  024644  012737  024754  001446                  MOV     #TST30,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
3697                                                                          ;TEST FOR ESCAPE ON PARITY ERRORS
3698  024652  012737  024670  001110  20$:     MOV     #11$,$LPERR     ;SET LOOP ON ERROR POINTER TO 11$
3699  024660  012702  000016                  MOV     #16,R2          ;PUT EXPECTED DATA IN R2
3700  024664  012700  177600                  MOV     #UIPDR0,R0      ;PUT ADDRESS OF REGISTER IN R0
3701  024670  005037  177600          11$:    CLR     UIPDR0          ;CLEAR THE REGISTER UNDER TEST
3702  024674  112710  172016                  MOVB    #172016,(R0)    ;LOAD LOWER BYTE OF REGISTER
3703  024700  013701  177600                  MOV     UIPDR0,R1       ;READ REGISTER INTO R1
```

```
3704  024704  020102                          CMP     R1,R2           ;SEE IF ONLY LOWER BYTE WAS WRITTEN
3705  024706  001401                          BEQ     1$              ;BRANCH IF DATA MATCHES
3706  024710  104017                          ERROR   17              ;DIDN'T LOAD CORRECT BYTE
3707  024712  012737  024730  001110  1$:     MOV     #12$,$LPERR     ;SET LOOP ON ERROR POINTER TO 12$
3708  024720  012700  177601                  MOV     #UIPDR0+1,R0    ;POINT TO UPPER BYTE OF REGISTER
3709  024724  012702  007416                  MOV     #007416,R2      ;LOAD EXPECTED DATA IN R2
3710  024730  112710  000017          12$:    MOVB    #17,(R0)        ;WRITE THE UPPER BYTE OF REGISTER
3711  024734  013701  177600                  MOV     UIPDR0,R1       ;READ REGISTER INTO R1
3712  024740  020102                          CMP     R1,R2           ;SEE IF REGISTER HOLDS CORRECT DATA
3713  024742  001401                          BEQ     2$              ;BRANCH TO EXIT IF DATA RIGHT
3714  024744  104017                          ERROR   17              ;DIDN'T LOAD CORRECT BYTE
3715
3716  024746  012737  024652  001110  2$:     MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
3717
3718
3719                                  .SBTTL          DUAL ADDRESSING BETWEEN GROUPS OF REGISTERS
3720                                  ;;*****************************************************************
3721                                  ;*TEST 30        DUAL ADDRESSING FOR ALL PAR'S AND PDR'S
3722                                  ;*
3723                                  ;*      THIS TEST WILL ENSURE THAT THERE IS NO DUAL ADDRESSING BETWEEN
3724                                  ;*      GROUPS OF PAR'S AND PDR'S.  THAT IS, WHEN YOU REFERENCE A KERNEL
3725                                  ;*      PAR YOU ARE REALLY REFERENCING THAT PAR.  FIRST EACH
3726                                  ;*      PAR0 OR PDR0 IS LOADED WITH A UNIQUE NUMBER 0-6 AND
3727                                  ;*      THEN THEY ARE EACH CHECKED FOR THE CORRECT DATA.
3728                                  ;*      EACH GROUP HAS ALREADY BEEN CHECKED FOR DUAL ADDRESSING WITHIN
3729                                  ;*      ITS OWN GROUP, SO ONLY ONE REGISTER FROM EACH GROUP NEEDS TO
3730                                  ;*      BE TESTED HERE.
3731                                  ;*
3732                                  ;;*****************************************************************
3733  024754                          TST30:
3734  024754  000004                          SCOPE
3735  024756  012737  025056  001446          MOV     #TST31,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
3736                                                                  ;TEST FOR ESCAPE ON PARITY ERRORS
3737  024764  012737  025000  001110  20$:     MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
3738  024772  005000                          CLR     R0              ;THIS WILL HOLD THE INDEX OF EACH REGISTER
3739                                                                  ;AND THE COUNT LOADED
3740  024774  012704  000004                  MOV     #4,R4           ;THIS IS THE NUMBER OF TIMES TO DO THIS LOOP
3741  025000  010070  001452          1$:     MOV     R0,@PARTAB(R0)  ;LOAD PAR OR PDR WITH INDEX NUMBER
3742  025004  062700  000002                  ADD     #2,R0           ;CHANGE INDEX TO POINT TO NEXT REGISTER
3743  025010  077405                          SOB     R4,1$           ;BRANCH BACK 3 TIMES
3744  025012  012704  000004                  MOV     #4,R4           ;DO NEXT LOOP 4 TIMES ALSO
3745  025016  012737  025030  001110          MOV     #10$,$LPERR     ;SET LOOP ON ERROR POINTER TO 10$
3746  025024  162700  000002          2$:     SUB     #2,R0           ;ADJUST INDEX FOR REGISTER DESIRED
3747  025030  017001  001452          10$:    MOV     @PARTAB(R0),R1  ;READ PAR OR PDR INTO R1
3748  025034  020001                          CMP     R0,R1           ;SEE IF INDEX EQUALS DATA
3749  025036  001403                          BEQ     3$              ;BRANCH IFCORRECT REGISTER WAS READ.
3750  025040  016002  001452                  MOV     PARTAB(R0),R2   ;SAVE ADDRESS OF BAD REGISTER
3751  025044  104030                          ERROR   30              ;DUAL ADDRESSING
3752  025046  077412                  3$:     SOB     R4,2$           ;BRANCH BACK 3 TIMES
3753  025050  012737  024764  001110          MOV     #20$,$LPERR     ;SET LOOP ON ERROR POINTER TO START OF TEST
3754
3755
3756                                  .SBTTL  ************ ENTRY POINT 4 --- STARTING ADDRESS 214 ************
3757                                  .SBTTL  ****************** RELOCATION AND ADDER TESTS ******************
3758                                  ;*
3759                                  ;*      THIS GROUP OF TESTS CHECKS MEMORY MANAGEMENT'S RELOCATION ADDER
```

```
                                        ;****************** RELOCATION AND ADDER TESTS ******************
                                        ;*      FIRST USING DESTINATION ONLY RELOCATION, THEN WITH FULL 18-BIT
                                        ;*      RELOCATION .
                                        ;*


                                        ;;*********************************************************************
                                        ;*TEST 31       18-BIT MAPPING ADDER TESTING
                                        ;*
                                        ;*      THIS TEST USES 'DESTINATION ONLY' RELOCATION TO CHECK THE
                                        ;*      FULL ADD PROPERTIES OF THE RELOCATION ADDER.  TWELVE PAIRS
                                        ;*      OF NUMBERS ARE ADDED, GENERATING PHYSICAL ADDRESSES ABOVE
                                        ;*      12K.
                                        ;*
                                        ;;*********************************************************************
025056                                  TST31:
                                                SCOPE
025056  000004
025060  012737  026620  001446                MOV     #TST32,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
                                                                        ;TEST FOR ESCAPE ON PARITY ERRORS
                                        ;
                                        ;       THE FOLLOWING CODE WILL CLEAR ALL PAR'S AND PDR'S SO THAT
                                        ;       THE RELOCATION ADDERS CAN BE CHECKED, ONLY THE KERNEL
                                        ;       PDR'S AND PAR'S WILL BE MAPPED RESIDENT AND READ WRITE.
                                        ;
025066                                  ENTPT4:
025066  012737  025150  001110                MOV     #20$,$LPERR     ;SET LOOP ON ERROR POINTER TO 20$
025074  012737  000031  001102                MOV     #31,$TSTNM      ;LOAD TEST NUMBER INTO MEMORY
025102  013777  001102  154032                MOV     $TSTNM,@DISPLAY ;DISPLAY TEST NUMBER FOR THIS TEST
025110  012705  172300                        MOV     #KIPDR0,R5      ;PUT ADDRESS OF KIPDR0 IN R5
025114  004737  044164                        JSR     PC,CLRREG       ;CLEAR KERNEL PDR'S
025120  012705  172340                        MOV     #KIPAR0,R5      ;PUT ADDRESS OF KIPAR0 IN R5
025124  004737  044164                        JSR     PC,CLRREG       ;CLEAR KERNEL PAR'S
025130  012705  177600                        MOV     #UIPDR0,R5      ;PUT ADDRESS OF UIPDR0 IN R5
025134  004737  044164                        JSR     PC,CLRREG       ;CLEAR USER PDR'S
025140  012705  177640                        MOV     #UIPAR0,R5      ;PUT ADDRESS OF UIPAR0 IN R5
025144  004737  044164                        JSR     PC,CLRREG       ;CLEAR USER PAR'S
025150  012700  077406                20$:     MOV     #77406,R0       ;MAKE KERNEL I PAGES 4K, R/W, EXPAND UP
025154  012702  000010                        MOV     #10,R2          ;SET COUNT TO LOAD 8 ADDRESSES
025160  012701  172300                        MOV     #KIPDR0,R1      ;PUT ADDRESS OF FIRST PDR IN R1
025164  010021                        64$:     MOV     R0,(R1)+        ;LOAD R0 INTO PDR ADDRESSED BY R1
025166  077202                                SOB     R2,64$          ;BRANCH BACK TO 64$ IF R2 IS NOT ZERO
025170  012737  000000  172340                MOV     #000,KIPAR0     ;MAP PAGE 0 TO 0 - 4K
025176  012737  000200  172342                MOV     #200,KIPAR1     ;MAP PAGE 1 TO 4K - 8K
025204  012737  000400  172344                MOV     #400,KIPAR2     ;MAP PAGE 2 TO 8K - 12K
025212  012737  000600  172346                MOV     #600,KIPAR3     ;MAP PAGE 3 TO 12K - 16K
025220  012737  007600  172356                MOV     #7600,KIPAR7    ;MAP PAGE 7 TO I/O PAGE
025226  012737  025274  001110                MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
025234  013737  060000  001176                MOV     @#60000,$TMP0   ;SAVE DATA AT TEST LOCATION
025242  012737  000600  172350                MOV     #600,@#KIPAR4   ;LOAD PAR4 WITH 600
025250  012737  000600  001200                MOV     #600,$TMP1      ;SAVE 600 FOR ERROR REPORT
025256  012737  060000  001202                MOV     #60000,$TMP2    ;SAVE 60000 FOR ERROR REPORT
025264  012700  100000                        MOV     #100000,R0      ;PUT VIRTUAL ADDRESS IN R0
025270  012701  125200                        MOV     #125200,R1      ;PUT DATA PATTERN IN R1
025274  052737  000400  177572        1$:     BIS     #BIT8,@#MMR0    ;TURN ON DESTINATION ONLY RELOCATION
025302  010110                                MOV     R1,(R0)         ;TRY TO LOAD DATA PATTERN INTO 60000
025304  013702  060000                        MOV     @#60000,R2      ;READ (60000) INTO R2
025310  000005                                RESET                   ;CLEAR MMR0
```

```
3816  025312  020102                        CMP     R1,R2           ;SEE IF DATA MATCHES PATTERN
3817  025314  001401                        BEQ     2$              ;BRANCH IF DATA MATCHES
3818  025316  104031                        ERROR   31              ;RELOCATION FAILED
3819  025320  013737  001176  060000  2$:   MOV     $TMP0,@#60000   ;RESTORE ORIGINAL DATA TO TEST LOCATION
3820  025326  012737  025374  001110        MOV     #7$,$LPERR      ;SET LOOP ON ERROR POINTER TO 7$
3821  025334  013737  062000  001176        MOV     @#62000,$TMP0   ;SAVE DATA AT TEST LOCATION
3822  025342  012737  000610  172350        MOV     #610,@#KIPAR4   ;LOAD PAR4 WITH 610
3823  025350  012737  000610  001200        MOV     #610,$TMP1      ;SAVE 610 FOR ERROR REPORT
3824  025356  012737  062000  001202        MOV     #62000,$TMP2    ;SAVE 62000 FOR ERROR REPORT
3825  025364  012700  101000                MOV     #101000,R0      ;PUT VIRTUAL ADDRESS IN R0
3826  025370  012701  125203                MOV     #125203,R1      ;PUT DATA PATTERN IN R1
3827  025374  052737  000400  177572  7$:   BIS     #BIT8,@#MMR0    ;TURN ON DESTINATION ONLY RELOCATION
3828  025402  010110                        MOV     R1,(R0)         ;TRY TO LOAD DATA PATTERN INTO 62000
3829  025404  013702  062000                MOV     @#62000,R2      ;READ (62000) INTO R2
3830  025410  000005                        RESET                   ;CLEAR MMR0
3831  025412  020102                        CMP     R1,R2           ;SEE IF DATA MATCHES PATTERN
3832  025414  001401                        BEQ     8$              ;BRANCH IF DATA MATCHES
3833  025416  104031                        ERROR   31              ;RELOCATION FAILED
3834  025420  013737  001176  062000  8$:   MOV     $TMP0,@#62000   ;RESTORE ORIGINAL DATA TO TEST LOCATION
3835  025426  012737  025474  001110        MOV     #9$,$LPERR      ;SET LOOP ON ERROR POINTER TO 9$
3836  025434  013737  062000  001176        MOV     @#62000,$TMP0   ;SAVE DATA AT TEST LOCATION
3837  025442  012737  000514  172350        MOV     #514,@#KIPAR4   ;LOAD PAR4 WITH 514
3838  025450  012737  000514  001200        MOV     #514,$TMP1      ;SAVE 514 FOR ERROR REPORT
3839  025456  012737  062000  001202        MOV     #62000,$TMP2    ;SAVE 62000 FOR ERROR REPORT
3840  025464  012700  110400                MOV     #110400,R0      ;PUT VIRTUAL ADDRESS IN R0
3841  025470  012701  125204                MOV     #125204,R1      ;PUT DATA PATTERN IN R1
3842  025474  052737  000400  177572  9$:   BIS     #BIT8,@#MMR0    ;TURN ON DESTINATION ONLY RELOCATION
3843  025502  010110                        MOV     R1,(R0)         ;TRY TO LOAD DATA PATTERN INTO 62000
3844  025504  013702  062000                MOV     @#62000,R2      ;READ (62000) INTO R2
3845  025510  000005                        RESET                   ;CLEAR MMR0
3846  025512  020102                        CMP     R1,R2           ;SEE IF DATA MATCHES PATTERN
3847  025514  001401                        BEQ     10$             ;BRANCH IF DATA MATCHES
3848  025516  104031                        ERROR   31              ;RELOCATION FAILED
3849  025520  013737  001176  062000  10$:  MOV     $TMP0,@#62000   ;RESTORE ORIGINAL DATA TO TEST LOCATION
3850  025526  012737  025574  001110        MOV     #11$,$LPERR     ;SET LOOP ON ERROR POINTER TO 11$
3851  025534  013737  062000  001176        MOV     @#62000,$TMP0   ;SAVE DATA AT TEST LOCATION
3852  025542  012737  000504  172350        MOV     #504,@#KIPAR4   ;LOAD PAR4 WITH 504
3853  025550  012737  000504  001200        MOV     #504,$TMP1      ;SAVE 504 FOR ERROR REPORT
3854  025556  012737  062000  001202        MOV     #62000,$TMP2    ;SAVE 62000 FOR ERROR REPORT
3855  025564  012700  111400                MOV     #111400,R0      ;PUT VIRTUAL ADDRESS IN R0
3856  025570  012701  125205                MOV     #125205,R1      ;PUT DATA PATTERN IN R1
3857  025574  052737  000400  177572  11$:  BIS     #BIT8,@#MMR0    ;TURN ON DESTINATION ONLY RELOCATION
3858  025602  010110                        MOV     R1,(R0)         ;TRY TO LOAD DATA PATTERN INTO 62000
3859  025604  013702  062000                MOV     @#62000,R2      ;READ (62000) INTO R2
3860  025610  000005                        RESET                   ;CLEAR MMR0
3861  025612  020102                        CMP     R1,R2           ;SEE IF DATA MATCHES PATTERN
3862  025614  001401                        BEQ     12$             ;BRANCH IF DATA MATCHES
3863  025616  104031                        ERROR   31              ;RELOCATION FAILED
3864  025620  013737  001176  062000  12$:  MOV     $TMP0,@#62000   ;RESTORE ORIGINAL DATA TO TEST LOCATION
3865  025626  012737  025674  001110        MOV     #13$,$LPERR     ;SET LOOP ON ERROR POINTER TO 13$
3866  025634  013737  062000  001176        MOV     @#62000,$TMP0   ;SAVE DATA AT TEST LOCATION
3867  025642  012737  000556  172350        MOV     #556,@#KIPAR4   ;LOAD PAR4 WITH 556
3868  025650  012737  000556  001200        MOV     #556,$TMP1      ;SAVE 556 FOR ERROR REPORT
3869  025656  012737  062000  001202        MOV     #62000,$TMP2    ;SAVE 62000 FOR ERROR REPORT
3870  025664  012700  104200                MOV     #104200,R0      ;PUT VIRTUAL ADDRESS IN R0
3871  025670  012701  125206                MOV     #125206,R1      ;PUT DATA PATTERN IN R1
```

```
3872  025674  052737  000400  177572  13$:    BIS     #BIT8,@#MMR0    ;TURN ON DESTINATION ONLY RELOCATION
3873  025702  010110                          MOV     R1,(R0)         ;TRY TO LOAD DATA PATTERN INTO 62000
3874  025704  013702  062000                  MOV     @#62000,R2      ;READ (62000) INTO R2
3875  025710  000005                          RESET                   ;CLEAR MMR0
3876  025712  020102                          CMP     R1,R2           ;SEE IF DATA MATCHES PATTERN
3877  025714  001401                          BEQ     14$             ;BRANCH IF DATA MATCHES
3878  025716  104031                          ERROR   31              ;RELOCATION FAILED
3879  025720  013737  001176  062000  14$:    MOV     $TMP0,@#62000   ;RESTORE ORIGINAL DATA TO TEST LOCATION
3880  025726  012737  025774  001110          MOV     #15$,$LPERR     ;SET LOOP ON ERROR POINTER TO 15$
3881  025734  013737  062000  001176          MOV     @#62000,$TMP0   ;SAVE DATA AT TEST LOCATION
3882  025742  012737  000442  172350          MOV     #442,@#KIPAR4   ;LOAD PAR4 WITH 442
3883  025750  012737  000442  001200          MOV     #442,$TMP1      ;SAVE 442 FOR ERROR REPORT
3884  025756  012737  062000  001202          MOV     #62000,$TMP2    ;SAVE 62000 FOR ERROR REPORT
3885  025764  012700  115600                  MOV     #115600,R0      ;PUT VIRTUAL ADDRESS IN R0
3886  025770  012701  125207                  MOV     #125207,R1      ;PUT DATA PATTERN IN R1
3887  025774  052737  000400  177572  15$:    BIS     #BIT8,@#MMR0    ;TURN ON DESTINATION ONLY RELOCATION
3888  026002  010110                          MOV     R1,(R0)         ;TRY TO LOAD DATA PATTERN INTO 62000
3889  026004  013702  062000                  MOV     @#62000,R2      ;READ (62000) INTO R2
3890  026010  000005                          RESET                   ;CLEAR MMR0
3891  026012  020102                          CMP     R1,R2           ;SEE IF DATA MATCHES PATTERN
3892  026014  001401                          BEQ     16$             ;BRANCH IF DATA MATCHES
3893  026016  104031                          ERROR   31              ;RELOCATION FAILED
3894  026020  013737  001176  062000  16$:    MOV     $TMP0,@#62000   ;RESTORE ORIGINAL DATA TO TEST LOCATION
3895  026026  012737  026074  001110          MOV     #17$,$LPERR     ;SET LOOP ON ERROR POINTER TO 17$
3896  026034  013737  062000  001176          MOV     @#62000,$TMP0   ;SAVE DATA AT TEST LOCATION
3897  026042  012737  000577  172350          MOV     #577,@#KIPAR4   ;LOAD PAR4 WITH 577
3898  026050  012737  000577  001200          MOV     #577,$TMP1      ;SAVE 577 FOR ERROR REPORT
3899  026056  012737  062000  001202          MOV     #62000,$TMP2    ;SAVE 62000 FOR ERROR REPORT
3900  026064  012700  102100                  MOV     #102100,R0      ;PUT VIRTUAL ADDRESS IN R0
3901  026070  012701  125210                  MOV     #125210,R1      ;PUT DATA PATTERN IN R1
3902  026074  052737  000400  177572  17$:    BIS     #BIT8,@#MMR0    ;TURN ON DESTINATION ONLY RELOCATION
3903  026102  010110                          MOV     R1,(R0)         ;TRY TO LOAD DATA PATTERN INTO 62000
3904  026104  013702  062000                  MOV     @#62000,R2      ;READ (62000) INTO R2
3905  026110  000005                          RESET                   ;CLEAR MMR0
3906  026112  020102                          CMP     R1,R2           ;SEE IF DATA MATCHES PATTERN
3907  026114  001401                          BEQ     18$             ;BRANCH IF DATA MATCHES
3908  026116  104031                          ERROR   31              ;RELOCATION FAILED
3909  026120  013737  001176  062000  18$:    MOV     $TMP0,@#62000   ;RESTORE ORIGINAL DATA TO TEST LOCATION
3910  026126  012737  026174  001110          MOV     #21$,$LPERR     ;SET LOOP ON ERROR POINTER TO 21$
3911  026134  013737  062000  001176          MOV     @#62000,$TMP0   ;SAVE DATA AT TEST LOCATION
3912  026142  012737  000421  172350          MOV     #421,@#KIPAR4   ;LOAD PAR4 WITH 421
3913  026150  012737  000421  001200          MOV     #421,$TMP1      ;SAVE 421 FOR ERROR REPORT
3914  026156  012737  062000  001202          MOV     #62000,$TMP2    ;SAVE 62000 FOR ERROR REPORT
3915  026164  012700  117700                  MOV     #117700,R0      ;PUT VIRTUAL ADDRESS IN R0
3916  026170  012701  125211                  MOV     #125211,R1      ;PUT DATA PATTERN IN R1
3917  026174  052737  000400  177572  21$:    BIS     #BIT8,@#MMR0    ;TURN ON DESTINATION ONLY RELOCATION
3918  026202  010110                          MOV     R1,(R0)         ;TRY TO LOAD DATA PATTERN INTO 62000
3919  026204  013702  062000                  MOV     @#62000,R2      ;READ (62000) INTO R2
3920  026210  000005                          RESET                   ;CLEAR MMR0
3921  026212  020102                          CMP     R1,R2           ;SEE IF DATA MATCHES PATTERN
3922  026214  001401                          BEQ     22$             ;BRANCH IF DATA MATCHES
3923  026216  104031                          ERROR   31              ;RELOCATION FAILED
3924  026220  013737  001176  062000  22$:    MOV     $TMP0,@#62000   ;RESTORE ORIGINAL DATA TO TEST LOCATION
3925  026226  012737  026274  001110          MOV     #23$,$LPERR     ;SET LOOP ON ERROR POINTER TO 23$
3926  026234  013737  060000  001176          MOV     @#60000,$TMP0   ;SAVE DATA AT TEST LOCATION
3927  026242  012737  000577  172350          MOV     #577,@#KIPAR4   ;LOAD PAR4 WITH 577
```

```
3928  026250  012737  000577  001200          MOV     #577,$TMP1      ;SAVE 577 FOR ERROR REPORT
3929  026256  012737  060000  001202          MOV     #60000,$TMP2    ;SAVE 60000 FOR ERROR REPORT
3930  026264  012700  100100                  MOV     #100100,R0      ;PUT VIRTUAL ADDRESS IN R0
3931  026270  012701  125212                  MOV     #125212,R1      ;PUT DATA PATTERN IN R1
3932  026274  052737  000400  177572  23$:    BIS     #BIT8,@#MMR0    ;TURN ON DESTINATION ONLY RELOCATION
3933  026302  010110                          MOV     R1,(R0)         ;TRY TO LOAD DATA PATTERN INTO 60000
3934  026304  013702  060000                  MOV     @#60000,R2      ;READ (60000) INTO R2
3935  026310  000005                          RESET                   ;CLEAR MMR0
3936  026312  020102                          CMP     R1,R2           ;SEE IF DATA MATCHES PATTERN
3937  026314  001401                          BEQ     24$             ;BRANCH IF DATA MATCHES
3938  026316  104031                          ERROR   31              ;RELOCATION FAILED
3939  026320  013737  001176  060000  24$:    MOV     $TMP0,@#60000   ;RESTORE ORIGINAL DATA TO TEST LOCATION
3940  026326  012737  026374  001110          MOV     #25$,$LPERR     ;SET LOOP ON ERROR POINTER TO 25$
3941  026334  013737  060000  001176          MOV     @#60000,$TMP0   ;SAVE DATA AT TEST LOCATION
3942  026342  012737  000570  172350          MOV     #570,@#KIPAR4   ;LOAD PAR4 WITH 570
3943  026350  012737  000570  001200          MOV     #570,$TMP1      ;SAVE 570 FOR ERROR REPORT
3944  026356  012737  060000  001202          MOV     #60000,$TMP2    ;SAVE 60000 FOR ERROR REPORT
3945  026364  012700  101000                  MOV     #101000,R0      ;PUT VIRTUAL ADDRESS IN R0
3946  026370  012701  125213                  MOV     #125213,R1      ;PUT DATA PATTERN IN R1
3947  026374  052737  000400  177572  25$:    BIS     #BIT8,@#MMR0    ;TURN ON DESTINATION ONLY RELOCATION
3948  026402  010110                          MOV     R1,(R0)         ;TRY TO LOAD DATA PATTERN INTO 60000
3949  026404  013702  060000                  MOV     @#60000,R2      ;READ (60000) INTO R2
3950  026410  000005                          RESET                   ;CLEAR MMR0
3951  026412  020102                          CMP     R1,R2           ;SEE IF DATA MATCHES PATTERN
3952  026414  001401                          BEQ     26$             ;BRANCH IF DATA MATCHES
3953  026416  104031                          ERROR   31              ;RELOCATION FAILED
3954  026420  013737  001176  060000  26$:    MOV     $TMP0,@#60000   ;RESTORE ORIGINAL DATA TO TEST LOCATION
3955                                  ;*
3956                                  ;*      THE FOLLOWING CODE PERFORMS A COUPLE RELOCATION ADDER CHECKS
3957                                  ;*      USING THE PROCESSOR STATUS WORD AS THE PHYSICAL ADDRESS
3958                                  ;*      WHICH IS GENERATED TO CHECK THE UPPER BITS OF THE "PHYSICAL
3959                                  ;*      BUS ADDRESS ALU".  SPECIAL DATA PATTERNS ARE USED DUE TO
3960                                  ;*      THE CONDITION CODE BITS.
3961
3962  026426  012737  026466  001110          MOV     #27$,$LPERR     ;SET LOOP ON ERROR POINTER TO 27$
3963  026434  012737  007600  172350          MOV     #7600,KIPAR4    ;LOAD PAR4 WITH 7600
3964  026442  012737  007600  001200          MOV     #7600,$TMP1     ;SAVE PAR4 DATA FOR ERROR REPORT
3965  026450  012737  177776  001202          MOV     #177776,$TMP2   ;SAVE PHYS. ADDR. FOR ERROR REPORT
3966  026456  012700  117776                  MOV     #117776,R0      ;PUT VIRTUAL ADDR. IN R0
3967  026462  012701  000100                  MOV     #100,R1         ;PUT DATA PATTERN IN R1
3968  026466  052737  000400  177572  27$:    BIS     #BIT8,@#MMR0    ;TURN ON DEST.-ONLY-RELOCATION
3969  026474  010110                          MOV     R1,(R0)         ;LOAD DATA INTO PSW USING PAR4
3970  026476  013702  177776                  MOV     @#177776,R2     ;READ PSW INTO R2
3971  026502  000005                          RESET                   ;CLEAR MMR0
3972  026504  042702  000020                  BIC     #20,R2          ;CLEAR T-BIT IF SET IN DATA READ
3973  026510  020102                          CMP     R1,R2           ;SEE IF DATA MATCHES PATTERN WRITTEN
3974  026512  001401                          BEQ     28$             ;BRANCH IF DATA MATCHES
3975  026514  104031                          ERROR   31              ;RELOCATION FAILED
3976  026516  012737  026556  001110  28$:    MOV     #29$,$LPERR     ;SET LOOP ON ERROR POINTER TO 29$
3977  026524  012737  007777  172350          MOV     #7777,KIPAR4    ;LOAD PAR4 WITH 7777
3978  026532  012737  007777  001200          MOV     #7777,$TMP1     ;SAVE PAR4 DATA FOR ERROR REPORT
3979  026540  012737  177776  001202          MOV     #177776,$TMP2   ;SAVE PHYS. ADDR. FOR ERROR REPORT
3980  026546  012700  100076                  MOV     #100076,R0      ;PUT VIRTUAL ADDR. R0
3981  026552  012701  000200                  MOV     #200,R1         ;PUT DATA PATTERN IN R1
3982  026556  052737  000400  177572  29$:    BIS     #BIT8,@#MMR0    ;TURN ON DEST.-ONLY-RELOCATION
3983  026564  010110                          MOV     R1,(R0)         ;LOAD DATA INTO PSW USING PAR4
```

```
3984  026566  013702  177776            MOV     @#177776,R2     ;READ PSW INTO R2
3985  026572  000005                    RESET                   ;CLEAR MMR0
3986  026574  042702  000020            BIC     #20,R2          ;CLEAR T-BIT IF SET IN DATA READ
3987  026600  020102                    CMP     R1,R2           ;SEE IF DATA MATCHES PATTERN
3988  026602  001401                    BEQ     30$             ;BRANCH IF DATA MATCHES
3989  026604  104031                    ERROR   31              ;RELOCATION FAILED
3990  026606                    30$:
3991  026606  005037  177776            CLR     @#PSW           ;CLEAR PSW BEFORE LEAVING TEST
3992  026612  012737  025150  001110    MOV     #20$,$LPERR     ;SET LOOP ON ERROR POINTER TO START OF TEST
3993
3994                              ;;*****************************************************************
3995                              ;*TEST 32       18-BIT MAPPING CARRY PROPAGATION
3996                              ;*
3997                              ;*      THIS TEST USES FULL 18-BIT RELOCATION TO CHECK THE CARRY
3998                              ;*      PROPAGATION OF THE RELOCATION ADDER.  SINCE THIS TEST SCANS
3999                              ;*      MEMORY FROM 060000 TO 750000 ON 2K BOUNDARIES, IT WILL
4000                              ;*      REPORT ANY HOLES THAT IT FINDS IN MEMORY UP TO THE 124K
4001                              ;*      THE INFORMATION GIVEN WILL BE THE ADDRESS WHERE
4002                              ;*      THE HOLE WAS DISCOVERED AND THE FIRST GOOD ADDRESS AFTER THE
4003                              ;*      HOLE.
4004                              ;;*****************************************************************
4005  026620                      TST32:
4006  026620  000004                    SCOPE
4007  026622  012737  027166  001446    MOV     #TST33,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
4008                                                            ;TEST FOR ESCAPE ON PARITY ERRORS
4009  026630  005037  001434    20$:    CLR     HOLFLG          ;MAKE SURE HOLE FLAG STARTS AT 0
4010  026634  005037  001200            CLR     $TMP1           ;CLEAR SPOT TO SAVE HOLE ADDRESS
4011  026640  012737  027156  000004    MOV     #11$,ERRVEC     ;SET ERROR POINTER TO 11$ FOR 11/6X
4012  026646  105737  001450            TSTB    FORTY           ;EXECUTING ON AN 11/40
4013  026652  001403                    BEQ     7$              ;BRANCH IF NO
4014  026654  012737  027146  000004    MOV     #10$,ERRVEC     ;SET ERROR POINTER TO 10$ FOR 11/40
4015  026662                    7$:
4016  026662  012737  000577  172350    MOV     #577,KIPAR4     ;LOAD PAR4 WITH STARTING BASE
4017  026670  012737  000600  172352    MOV     #600,KIPAR5     ;START ADDRESSING WITH 12K
4018  026676  012700  100100            MOV     #100100,R0      ;LOAD VIRTUAL ADDR FOR PAR4 IN R0
4019  026702  012701  120000            MOV     #120000,R1      ;LOAD VIRTUAL ADDR FOR PAR5 IN R1
4020  026706  012702  001000            MOV     #1000,R2        ;LOAD DATA PATTERN INTO R2
4021  026712  012737  000001  177572    MOV     #1,@#MMR0       ;TURN ON 18-BIT MAPPING
4022                                    ;
4023                              ;FULL RELOCATION STARTS HERE AND CONTINUES FOR REST OF PROGRAM
4024                              ;
4025  026720  012737  027016  001110  1$:    MOV     #3$,$LPERR      ;SET LOOP ON ERROR POINTER TO 3$
4026  026726  005037  001410            CLR     PCPUER          ;CLEAR CPU ERROR FLAG
4027  026732  011137  001176            MOV     (R1),$TMP0      ;SAVE DATA AT TEST LOCATION
4028  026736  032737  000020  001410    BIT     #BIT4,PCPUER    ;SEE IF THERE WAS A CPU TRAP
4029  026744  001411                    BEQ     2$              ;BRANCH IF NO TIMEOUT
4030  026746  005237  001434            INC     HOLFLG          ;SET FLAG IF THERE WAS TIMEOUT
4031  026752  005737  001200            TST     $TMP1           ;HAVE I SAVED STARTING ADDR. OF HOLE
4032  026756  001034                    BNE     5$              ;  IF I HAVE I'LL CONTINUE
4033  026760  013737  172352  001200    MOV     KIPAR5,$TMP1    ;SAVE PAR-POSSIBLE START OF HOLE
4034  026766  000430                    BR      5$              ;BRANCH TO CHANGE BASE ADDRESS
4035  026770  005737  001434    2$:     TST     HOLFLG          ;IF HOLFLG CLEAR-NO TIMEOUTS YET
4036  026774  001410                    BEQ     3$              ;OTHERWISE HAVE FOUND MEMORY HOLE
4037  026776  013737  172352  001202    MOV     KIPAR5,$TMP2    ;SAVE PAR THAT POINTS TO END OF HOLE
4038  027004  104032                    ERROR   32              ;HOLE IN MEMORY FROM $TMP1 TO $TMP2
4039  027006  005037  001434            CLR     HOLFLG          ;CLEAR HOLE FLAG IN CASE THERES MORE
```

```
4040  027012  005037  001200                  CLR     $TMP1           ;CLEAR STARTING ADDR. LOC. ALSO
4041  027016  010210                  3$:     MOV     R2,(R0)         ;LOAD TEST PATTERN INTO TEST LOCATION
4042  027020  011103                          MOV     (R1),R3         ;READ TEST LOCATION VIA DIFFERENT PAR
4043  027022  020203                          CMP     R2,R3           ;SEE IF CORRECT LOCATION REFERENCED
4044  027024  001407                          BEQ     4$              ;BRANCH IF CORRECT DATA OBTAINED
4045  027026  013737  172350  001204          MOV     KIPAR4,$TMP3    ;SAVE PAR4 FOR ERROR REPORT
4046  027034  013737  172352  001206          MOV     KIPAR5,$TMP4    ;SAVE PAR5 FOR ERROR REPORT
4047  027042  104033                          ERROR   33              ;OTHERWISE BAD RELOCATION
4048  027044  013711  001176          4$:     MOV     $TMP0,(R1)      ;RESTORE ORIGINAL DATA TO TEST LOCATION
4049  027050  062737  000100  172350  5$:     ADD     #100,KIPAR4     ;CHANGE BASE ADDRESS
4050  027056  062737  000100  172352          ADD     #100,KIPAR5     ;CHANGE BASE ADDRESS
4051  027064  005202                          INC     R2              ;CHANGE DATA PATTERN
4052  027066  022737  007500  172352          CMP     #7500,KIPAR5    ;SEE IF PAST 122K-LAST ADDRESS
4053  027074  103311                          BHIS    1$              ;BRANCH IF NOT PAST LAST ADDRESS
4054                                  ;*
4055                                  ;*      ADDRESS 000000 IS GENERATED BY CARRY PROPAGATION, USING
4056                                  ;*      '7777' IN KIPAR4 WITH A VIRTUAL ADDRESS OF '100100'.
4057                                  ;*
4058                                  ;*
4059                                  ;*
4060                                  ;*
4061                                  ;*
4062
4063  027076  012737  027116  001110  6$:     MOV     #16$,$LPERR     ;SET LOOP ON ERROR POINTER TO 16$
4064  027104  012737  007777  172350          MOV     #7777,KIPAR4    ;LOAD PAR 4 WITH HIGHEST VALUE POSSIBLE
4065  027112  005037  000000                  CLR     @#000000        ;CLEAR ADDRESS ZERO
4066  027116  013701  100100          16$:    MOV     @#100100,R1     ;THIS SHOULD READ ADDRESS ZERO INTO R1
4067  027122  005701                          TST     R1              ;SEE IF YOU REALLY READ ADDRESS ZERO
4068  027124  001401                          BEQ     8$              ;BRANCH IF YOU READ ADDRESS ZERO
4069  027126  104034                          ERROR   34              ;DIDN'T READ ADDRESS ZERO
4070  027130  012737  015162  000004  8$:     MOV     #CPUER,ERRVEC   ;RESTORE NORMAL CPU TRAP ROUTINE
4071  027136  012737  026630  001110          MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
4072  027144  000410                          BR      TST33           ;;BRANCH TO NEXT TEST
4073
4074                                  ;;********************  TRAP TO HERE THRU ERRVEC  **********************
4075
4076  027146  012737  000020  001410  10$:     MOV     #BIT4,PCPUER    ;SET "TRAP-TO-4" FLAG
4077  027154  000002                          RTI
4078  027156  017737  152172  001410  11$:     MOV     @CPUERR,PCPUER  ;SAVE CPU ERROR REGISTER
4079  027164  000002                          RTI                     ;RETURN TO TEST AND CONTINUE
4080
4081
4082
4083
4084
4085
4086                                  .SBTTL  ************ ENTRY POINT 5 --- STARTING ADDRESS 220 ************
4087                                  .SBTTL  ******** MEMORY MANAGEMENT ABORTS LOGIC TESTS ********
4088                                  ;*
4089                                  ;*      THIS GROUP OF TESTS CHECKS OUT THE MEMORY MANAGEMENT ABORT LOGIC, AND
4090                                  ;*      ALSO CHECK OUT BITS <06:05> AND <03:00> OF MMR0, AND ALL BITS OF MMR2.
4091
4092
4093                                  ;;======================================================================
4094                                  ;*TEST 33       PAGE LENGTH FAULTS - UPWARD EXPANSION
4095                                  ;*
```

```
                                        ;*      THIS TEST CHECKS OUT THE PAGE LENGTH COMPARATORS
                                        ;*      AND THE LOGIC THAT GENERATES "PAGE LTH ERR".  IT TRIES
                                        ;*      EVERY PAGE LENGTH FIELD FROM 1 BLOCK TO 200(8) BLOCKS.  THE
                                        ;*      TEST THEN TRIES A REFERENCE AT EVERY 100 BYTES UNTIL AN ABORT
                                        ;*      OCCURS.  IF THE ABORT HAPPENS TOO SOON OR IF NO ABORT HAPPENS AT
                                        ;*      THE CORRECT BLOCK NUMBER AN ERROR IS REPORTED.
                                        ;*
                                        ;**********************************************************************
027166                                  TST33:
027166  000004                          SCOPE
027170  012737  027512  001446          MOV     #TST34,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
                                                                ;TEST FOR ESCAPE ON PARITY ERRORS
027176  012737  000024  001212          MOV     #24,$TIMES      ;;DO 24 ITERATIONS
027204                                  ENTPT5:
027204  012737  027330  001110          MOV     #20$,$LPERR     ;SET LOOP ON ERROR POINTER TO 20$
027212  012737  000033  001102          MOV     #33,$TSTNM      ;LOAD TEST NUMBER INTO MEMORY
027220  013777  001102  151714          MOV     $TSTNM,@DISPLAY ;DISPLAY TEST NUMBER FOR THIS TEST
027226  012737  077406  172300          MOV     #77406,KIPDR0   ;MAKE KERNEL I PAGE 0 200 BLOCKS, R/W
027234  012737  077406  172302          MOV     #77406,KIPDR1   ;MAKE KERNEL I PAGE 1 200 BLOCKS, R/W
027242  012737  077406  172304          MOV     #77406,KIPDR2   ;MAKE KERNEL I PAGE 2 200 BLOCKS, R/W
027250  012737  077406  172306          MOV     #77406,KIPDR3   ;MAKE KERNEL I PAGE 3 200 BLOCKS, R/W
027256  012737  077406  172316          MOV     #77406,KIPDR7   ;MAKE KERNEL I PAGE 7 200 BLOCKS, R/W
027264  012737  000000  172340          MOV     #000,KIPAR0     ;MAP KERNEL I PAGE 0 TO 0 - 4K
027272  012737  000200  172342          MOV     #200,KIPAR1     ;MAP KERNEL I PAGE 1 TO 4K - 8K
027300  012737  000400  172344          MOV     #400,KIPAR2     ;MAP KERNEL I PAGE 2 TO 8K - 12K
027306  012737  000600  172346          MOV     #600,KIPAR3     ;MAP KERNEL I PAGE 3 TO 12K - 16K
027314  012737  007600  172356          MOV     #7600,KIPAR7    ;MAP KERNEL I PAGE 7 TO THE I/O PAGE
027322  012737  000001  177572          MOV     #BIT0,MMR0      ;ENABLE 18-BIT RELOCATION IF NOT ON
027330  012737  000006  172310  20$:    MOV     #000006,@#KIPDR4 ;LOAD PDR4 FOR PAGE LENGTH OF 1
027336  012737  000600  172350          MOV     #600,KIPAR4     ;MAP PAGE 4 TO 12K
027344  012700  172311                  MOV     #KIPDR4+1,R0    ;PUT ADDRESS OF PDR 4'S UPPER BYTE IN R0
027350  012737  027412  001110          MOV     #3$,$LPERR      ;SET LOOP ON ERROR POINTER TO 3$
027356  005037  001404          1$:     CLR     PMMR0           ;CLEAR LOCATION THAT HOLDS MMR0
027362  012702  100000                  MOV     #100000,R2      ;PUT VIRTUAL ADDRESS INTO R2
027366  010203                  2$:     MOV     R2,R3           ;PUT VIRTUAL ADDRESS INTO R3 TOO
027370  006203                          ASR     R3              ;RIGHT SHIFT 6 BITS SO IT WILL
027372  006203                          ASR     R3              ;MATCH THE PLF
027374  006203                          ASR     R3
027376  006203                          ASR     R3
027400  006203                          ASR     R3
027402  006203                          ASR     R3
027404  042703  177600                  BIC     #177600,R3      ;CLEAR BITS THAT ARE SET IN UPPER BYTE
027410  111001                          MOVB    (R0),R1         ;SAVE PLF IN R1 FOR LATER COMPARISON
027412  012737  040011  001366  3$:     MOV     #040011,MMEXP   ;POTENTIAL ABORT CONDITION: PAGE LENGTH
                                                                ;ERROR KERNEL, I-SPACE, PAGE 4
027420  011204                          MOV     (R2),R4         ;READ USING V.A. IN R2
027422  005037  001366                  CLR     MMEXP           ;CLEAR EXPECTED ABORT CONDITION
027426  005737  001404                  TST     PMMR0           ;SEE IF ABORT OCCURRED YET
027432  001405                          BEQ     4$              ;BRANCH IF NO ABORT YET, CHANGE V.A.
027434  005303                          DEC     R3              ;MAKE R3 EQUAL TO PLF
027436  020103                          CMP     R1,R3           ;SEE IF PDR 4'S PLF=R3
027440  001414                          BEQ     6$              ;BRANCH IF ABORT HAPPENS AT RIGHT PLACE
027442  104035                          ERROR   35              ;ABORT WRONG PLACE
027444  000412                          BR      6$              ;BRANCH TO CHANGE PLF
027446  020103                  4$:     CMP     R1,R3           ;SEE IF PAGE LENGTH FAULT SHOULD HAVE OCCURRED
027450  103002                          BHIS    5$              ;BRANCH IF NO PAGE LENGTH FAULT CONDITION
```

```
 4152  027452  104036                          ERROR   36              ;NO ABORT, IT SHOULD HAVE HAPPENED THIS TIME
 4153  027454  000406                          BR      6$              ;BRANCH TO CHANGE PLF
 4154  027456  120327  000177          5$:     CMPB    R3,#177         ;SEE IF V.A. IS 177
 4155  027462  001403                          BEQ     6$              ;BRANCH TO CHECK PLF
 4156  027464  062702  000100                  ADD     #100,R2         ;CHANGE VIRTUAL ADDRESS
 4157  027470  000736                          BR      2$              ;GO TRY THE NEXT VIRTUAL ADDRESS
 4158  027472  122710  000177          6$:     CMPB    #177,(R0)       ;SEE IF PDR 4'S PLF IS 177
 4159  027476  001402                          BEQ     7$              ;BRANCH TO EXIT IF PLF IS 177
 4160  027500  105210                          INCB    (R0)            ;STEP PLF OF PDR 4 UP BY 1
 4161  027502  000725                          BR      1$              ;BRANCH TO START WITH V.A. OF 0
 4162  027504  012737  027330  001110  7$:     MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
 4163
 4164                                  ;;*****************************************************************
 4165                                  ;*TEST 34       PAGE LENGTH FAULTS - DOWNWARD EXPANSION
 4166                                  ;*
 4167                                  ;*      THIS TEST CHECKS OUT THE PAGE LENGTH COMPARATORS
 4168                                  ;*      AND THE LOGIC THAT GENERATES "PAGE LTH ERR".  IT TRIES
 4169                                  ;*      EVERY PAGE LENGTH FIELD FROM 1 BLOCK TO 200(8) BLOCKS.  THE
 4170                                  ;*      TEST THEN TRIES A REFERENCE AT EVERY 100 BYTES UNTIL AN ABORT
 4171                                  ;*      OCCURS.  IF THE ABORT HAPPENS TOO SOON OR IF NO ABORT HAPPENS AT
 4172                                  ;*      THE CORRECT BLOCK NUMBER AN ERROR IS REPORTED.
 4173                                  ;*
 4174                                  ;;*****************************************************************
 4175  027512                          TST34:
 4176  027512  000004                          SCOPE
 4177  027514  012737  027712  001446          MOV     #TST35,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
 4178                                                                  ;TEST FOR ESCAPE ON PARITY ERRORS
 4179  027522  012737  000024  001212          MOV     #24,$TIMES      ;;DO 24 ITERATIONS
 4180  027530  012737  077416  172310  20$:     MOV     #77416,@#KIPDR4 ;LOAD PDR4 FOR PAGE LENGTH OF 1
 4181  027536  012737  000600  172350          MOV     #600,KIPAR4     ;MAP PAGE 4 TO 12K
 4182  027544  012700  172311                  MOV     #KIPDR4+1,R0    ;PUT ADDRESS OF PDR 4'S UPPER BYTE IN R0
 4183  027550  012737  027612  001110          MOV     #3$,$LPERR      ;SET LOOP ON ERROR POINTER TO 3$
 4184  027556  005037  001404          1$:     CLR     PMMR0           ;CLEAR LOCATION THAT HOLDS MMR0
 4185  027562  012702  117700                  MOV     #117700,R2      ;PUT VIRTUAL ADDRESS INTO R2
 4186  027566  010203                  2$:     MOV     R2,R3           ;PUT VIRTUAL ADDRESS INTO R3 TOO
 4187  027570  006203                          ASR     R3              ;RIGHT SHIFT 6 BITS SO IT WILL
 4188  027572  006203                          ASR     R3              ;MATCH THE PLF
 4189  027574  006203                          ASR     R3
 4190  027576  006203                          ASR     R3
 4191  027600  006203                          ASR     R3
 4192  027602  006203                          ASR     R3
 4193  027604  042703  177600                  BIC     #177600,R3      ;CLEAR BITS THAT ARE SET IN UPPER BYTE
 4194  027610  111001                          MOVB    (R0),R1         ;SAVE PLF IN R1 FOR LATER COMPARISON
 4195  027612  012737  040011  001366  3$:     MOV     #040011,MMEXP   ;POTENTIAL ABORT CONDITION: PAGE LENGTH
 4196                                                                  ;ERROR KERNEL, I-SPACE, PAGE 4
 4197  027620  011204                          MOV     (R2),R4         ;READ USING V.A. IN R2
 4198  027622  005037  001366                  CLR     MMEXP           ;CLEAR EXPECTED ABORT CONDITION
 4199  027626  005737  001404                  TST     PMMR0           ;SEE IF ABORT OCCURRED YET
 4200  027632  001405                          BEQ     4$              ;BRANCH IF NO ABORT YET, CHANGE V.A.
 4201  027634  005203                          INC     R3              ;MAKE R3 EQUAL TO PLF
 4202  027636  020103                          CMP     R1,R3           ;SEE IF PDR 4'S PLF=R3
 4203  027640  001414                          BEQ     6$              ;BRANCH IF ABORT HAPPENS AT RIGHT PLACE
 4204  027642  104035                          ERROR   35              ;ABORT WRONG PLACE
 4205  027644  000412                          BR      6$              ;BRANCH TO CHANGE PLF
 4206  027646  020103                  4$:     CMP     R1,R3           ;SEE IF PAGE LENGTH FAULT SHOULD HAVE OCCURRED
 4207  027650  101402                          BLOS    5$              ;BRANCH IF NO PAGE LENGTH FAULT CONDITION
```

```
4208  027652  104036                        ERROR   36              ;NO ABORT, IT SHOULD HAVE HAPPENED THIS TIME
4209  027654  000406                        BR      6$              ;BRANCH TO CHANGE PLF
4210  027656  120327  000000        5$:     CMPB    R3,#000         ;SEE IF V.A. IS 000
4211  027662  001403                        BEQ     6$              ;BRANCH TO CHECK PLF
4212  027664  162702  000100                SUB     #100,R2         ;CHANGE VIRTUAL ADDRESS
4213  027670  000736                        BR      2$              ;GO TRY THE NEXT VIRTUAL ADDRESS
4214  027672  122710  000000        6$:     CMPB    #000,(R0)       ;SEE IF PDR 4'S PLF IS 000
4215  027676  001402                        BEQ     7$              ;BRANCH TO EXIT IF PLF IS 000
4216  027700  105310                        DECB    (R0)            ;STEP PLF OF PDR 4 UP BY 1
4217  027702  000725                        BR      1$              ;BRANCH TO START WITH V.A. OF 0
4218  027704  012737  027530  001110  7$:   MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
4219
4220
4221                                  ;;*****************************************************************
4222                                  ;*TEST 35       ACCESS CONTROL FIELD = 0, OR 4 (ABORT ALL ACCESSES)
4223                                  ;*
4224                                  ;*      THESE A.C.F.'S ARE ALL NON-RESIDENT, ANY REFERENCE (READ
4225                                  ;*      OR WRITE) TO A NON-RESIDENT PAGE SHOULD SET BIT 15 IN MMR0.
4226                                  ;*      BITS <06:05> AND <03:00> IN MMR0 WILL LOCK UP ALL AVAILABLE INFORMATION
4227                                  ;*      ON THAT PAGE.  IN THIS CASE THE PAGE THAT CAUSES THE ABORT
4228                                  ;*      IS KERNEL PAGE 5.  THE EXPECTED ERROR CODE IS 100013.
4229                                  ;*
4230                                  ;;*****************************************************************
4231  027712                          TST35:
4232  027712  000004                        SCOPE
4233  027714  012737  030210  001446        MOV     #TST36,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
4234                                                                ;TEST FOR ESCAPE ON PARITY ERRORS
4235  027722  012737  077406  172310  20$:   MOV     #77406,@#KIPDR4 ;LOAD ACF 6 INTO PDR 4
4236  027730  012737  077400  172312        MOV     #77400,@#KIPDR5 ;LOAD ACF 0 INTO PDR5
4237
4238  027736  012737  000600  172350        MOV     #600,@#KIPAR4   ;LOAD 12K INTO PAR4
4239  027744  012737  000600  172352        MOV     #600,@#KIPAR5   ;LOAD 12K INTO PAR5
4240  027752  012700  100000                MOV     #100000,R0      ;LOAD VIRTUAL ADDRESS FOR PAR4 INTO R0
4241  027756  012701  120000                MOV     #120000,R1      ;LOAD VIRTUAL ADDRESS FOR PAR5 INTO R1
4242  027762  012702  161457                MOV     #161457,R2      ;LOAD DATA PATTERN INTO R2
4243  027766  012737  027776  001110  11$:   MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
4244  027774  005010                        CLR     (R0)            ;CLEAR MEMORY LOCATION 60000
4245  027776  012737  100013  001366  1$:    MOV     #100013,MMEXP   ;LOAD EXPECTED ABORT CONDITION: NON-RESIDENT,
4246                                                                ;KERNEL, PAGE 5, FULL RELOCATION
4247  030004  010211                  8$:    MOV     R2,(R1)         ;WRITE TO NON-RESIDENT OR UNUSED ACF
4248                                                                ;SHOULD CAUSE ABORT
4249  030006  005037  001366                CLR     MMEXP           ;CLEAR EXPECTED ABORT CONDITION
4250
4251  030012  005710                        TST     (R0)            ;SEE IF (60000) IS STILL ZERO
4252  030014  001411                        BEQ     2$              ;BRANCH IF (60000) IS ZERO
4253  030016  013737  172312  001176        MOV     KIPDR5,$TMP0    ;SAVE KIPDR5 FOR ERROR REPORT
4254  030024  013737  172352  001200        MOV     KIPAR5,$TMP1    ;SAVE KIPAR5 FOR ERROR REPORT
4255  030032  010137  001202                MOV     R1,$TMP2        ;SAVE VIRT ADR. FOR ERROR REPORT
4256  030036  104037                        ERROR   37              ;ABORT DID NOT HAPPEN
4257  030040  012737  030052  001110  2$:    MOV     #4$,$LPERR      ;SET LOOP ON ERROR POINTER TO 4$
4258  030046  010210                        MOV     R2,(R0)         ;LOAD DATA PATTERN INTO 60000
4259  030050  005004                        CLR     R4              ;CLEAR REGISTER TO RECEIVE DATA
4260  030052  012737  100013  001366  4$:    MOV     #100013,MMEXP   ;LOAD EXPECTED ABORT CONDITION: NON-RESIDENT,
4261                                                                ;KERNEL, I-SPACE, PAGE 5, FULL RELOCATION
4262  030060  011104                  9$:    MOV     (R1),R4         ;TRY TO READ (100000) INTO R4
4263                                                                ;THIS SHOULD ABORT
```

```
4264  030062  005037  001366                    CLR     MMEXP           ;EXPECTED ABORT CONDITION
4265
4266  030066  005704                            TST     R4              ;MAKE SURE R4 IS STILL 0
4267  030070  001411                            BEQ     5$              ;BRANCH IF R4 IS 0
4268  030072  013737  172312  001176            MOV     KIPDR5,$TMP0    ;SAVE KIPDR5 FOR ERROR REPORT
4269  030100  013737  172352  001200            MOV     KIPAR5,$TMP1    ;SAVE KIPAR5 FOR ERROR REPORT
4270  030106  010137  001202                    MOV     R1,$TMP2        ;SAVE VIRT ADR. FOR ERROR REPORT
4271  030112  104037                            ERROR   37              ;ABORT DID NOT HAPPEN
4272  030114                          5$:
4273  030114  023727  172312  077404            CMP     KIPDR5,#77404   ;SEE IF PDR 5'S ACF=4
4274  030122  001404                            BEQ     12$             ;TEST OVER IF ACF=4
4275  030124  012737  077404  172312    10$:    MOV     #77404,KIPDR5   ;MAKE PDR 5'S ACF=4
4276  030132  000715                            BR      11$             ;REPEAT TEST WITH ACF=4
4277                                            ;;
4278                                            ;;  THIS SECTION OF CODE VERIFIES THAT YOU WON'T MODIFY A MEMORY
4279                                            ;;  MANAGEMENT REGISTER ON A M.M. ABORT REFERENCE.
4280                                            ;;
4281  030134  012737  030162  001110    12$:    MOV     #13$,$LPERR     ;SET LOOP ON ERROR POINTER TO 13$
4282  030142  012737  007600  172352            MOV     #7600,KIPAR5    ;MAP PAGE 5 TO I/O PAGE
4283  030150  012701  132350                    MOV     #132350,R1      ;LOAD VIRTUAL ADDRESS TO REFERENCE
4284                                                                    ;KIPAR4
4285  030154  012737  100013  001366            MOV     #100013,MMEXP   ;EXPECTING NON-RESIDENT ABORT
4286                                                                    ;KERNEL I PAGE 5
4287  030162  005011                    13$:    CLR     (R1)            ;THIS INSTRUCTION SHOULD ABORT
4288                                                                    ;DURING THE ABORT 'NON RES ERR' IS
4289                                                                    ;ASSERTED TO STOP THE CLOCKING OF THE
4290                                                                    ;FLIP/FLOPS
4291  030164  022737  000600  172350            CMP     #600,KIPAR4     ;KIPAR4 SHOULD STILL BE 600
4292  030172  001401                            BEQ     14$             ;BRANCH IF KIPAR4 IS STILL 600
4293  030174  104074                            ERROR   74              ;ABORT DID NOT STOP CLRING OF KIPAR4
4294  030176  012737  027722  001110    14$:    MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
4295  030204  005037  001366                    CLR     MMEXP           ;NOT EXPECTING ANY TRAPS
4296
4297
4298                                            ;;*****************************************************************
4299                                            ;*TEST 36       ACCESS CONTROL FIELD = 2  (ABORT ON WRITE)
4300                                            ;*
4301                                            ;*      THIS IS A READ ONLY A.C.F., ANY WRITE ATTEMPT TO THIS PAGE
4302                                            ;*      WILL ABORT AND SET BIT 13 OF MMR0.
4303                                            ;*      BITS <06:05> AND <03:01> IN MMR0 WILL LOCK UP ALL AVAILABLE INFORMATION
4304                                            ;*      ON THAT PAGE.  IN THIS CASE THE WRITE ATTEMPT WILL BE TO
4305                                            ;*      KERNEL SPACE PAGE 4.  THE EXPECTED ERROR CODE IS 020011.
4306                                            ;*
4307                                            ;;*****************************************************************
4308  030210                            TST36:
4309  030210  000004                            SCOPE
4310  030212  012737  030454  001446            MOV     #TST37,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
4311                                                                    ;TEST FOR ESCAPE ON PARITY ERRORS
4312  030220                            20$:
4313  030220  012737  000600  172352            MOV     #600,KIPAR5     ;MAP PAGE 5 TO 12K
4314  030226  012737  000600  172350            MOV     #600,KIPAR4     ;MAP PAGE 4 TO 12K
4315  030234  012737  077406  172312            MOV     #77406,KIPDR5   ;PAGE 5 IS 200 BLOCKS LONG,
4316                                                                    ;EXPANDS UPWARD, AND IS READ/WRITE
4317                                                                    ;WITH NO TRAPPING
4318  030242  012737  077402  172310            MOV     #77402,KIPDR4   ;LOAD ACF 2 INTO PDR 4
4319  030250  005037  001366                    CLR     MMEXP           ;NOT EXPECTING ANY TRAPS OR ABORTS YET
```

```
4320  030254  012737  002222  120000          MOV     #2222,@#120000  ;LOAD DATA INTO 12K
4321  030262  013700  100000                  MOV     @#100000,R0     ;READ DATA THRU PAGE 4
4322  030266  005037  120000                  CLR     @#120000        ;CLEAR TEST LOCATION THRU PAGE 5
4323  030272  012737  020011  001366          MOV     #20011,MMEXP    ;EXPECTING READ ONLY FAULT, PAGE 4
4324  030300  012737  017777  100000          MOV     #17777,@#100000 ;TRY TO WRITE THRU PAGE 4
4325  030306  005037  001366                  CLR     MMEXP           ;NO MORE TRAPS EXPECTED
4326  030312  005737  120000                  TST     @#120000        ;SEE IF TEST LOCATION IS STILL ZERO
4327  030316  001412                          BEQ     2$              ;BRANCH IF WORD IS STILL ZERO
4328  030320  013737  172310  001176          MOV     KIPDR4,$TMP0    ;SAVE KIPDR4 FOR ERROR REPORT
4329  030326  013737  172350  001200          MOV     KIPAR4,$TMP1    ;SAVE KIPAR4 FOR ERROR REPORT
4330  030334  012737  100000  001202          MOV     #100000,$TMP2   ;SAVE VIRT ADR. FOR ERROR REPORT
4331  030342  104040                          ERROR   40              ;NO ABORT ON PAGE 4
4332  030344  105737  001450          2$:     TSTB    FORTY           ;EXECUTING ON AN 11/40
4333  030350  001041                          BNE     5$              ;BRANCH IF  YES
4334  030352  076600                          MED                     ;READ THE LOG JAM REG.
4335  030354  000100                          RLJAM
4336  030356  030027  000020                  BIT     R0,#BIT4        ;WAS BIT 4 SET IN LOG JAM?
4337  030362  001007                          BNE     4$              ;BRANCH IF YES
4338  030364  012737  000020  001210          MOV     #BIT4,$TMP5     ;SAVE EXPECTED CONTENTS FOR ERROR
4339  030372  012737  000100  001206          MOV     #RLJAM,$TMP4    ;SAVE LOG JAM ADDRESS FOR ERROR
4340  030400  104022                          ERROR   22              ;BIT4 NOT SET IN LOG JAM
4341                                                                  ;TO INDICATE KT ABORT
4342  030402                          4$:
4343  030402  012737  030414  000004          MOV     #3$,ERRVEC      ;SET TRAP TO 4 VECTOR TO 3$
4344  030410  005737  000001                  TST     @#000001        ;CAUSE TRAP TO 4 WITH ODD ADDR. ERROR
4345  030414  022626                  3$:     CMP     (SP)+,(SP)+     ;CLEAN UP STACK
4346  030416  012737  001354  000004          MOV     #CPUERR,ERRVEC  ;RESTORE NORMAL CPU TRAP ROUTINE
4347                                                                  ;NOW CHECK FOR 250 AS LAST VEC. LOGGED
4348  030424  076600                          MED                     ;READ LOG FLAG/INTRPT REG.
4349  030426  000104                          RLFGIN
4350  030430  030027  000250                  BIT     R0,#250         ;WAS VECTOR=250 RECORDED
4351  030434  001007                          BNE     5$              ;BRANCH IF YES
4352  030436  012737  000250  001210          MOV     #250,$TMP5      ;SAVE EXPECTED CONTENTS FOR ERROR
4353  030444  012737  000104  001206          MOV     #RLFGIN,$TMP4   ;SAVE LOG FLAG INT ADDRESS FOR ERROR
4354  030452  104022                          ERROR   22              ;VECTOR 250 WAS NOT LOGGIN IN
4355  030454                          5$:                             ;LOG FLAG/INTERRUPT REG.
4356
4357
4358                                  ;;*****************************************************************
4359                                  ;*TEST 37       RED & YELLOW ZONE STACK VIOLATIONS
4360                                  ;*
4361                                  ;*      THE FOLLOWING TEST CHECKS THE 'RED ZONE' AND 'YELLOW ZONE' LOGIC
4362                                  ;*      ON PAGE K506.  BITS <15:08> OF THE STACK LIMIT REGISTER AND
4363                                  ;*      THE STACK POINTER ARE INCREMENTED FROM ZERO UP TO THE LAST
4364                                  ;*      EXSISTING MEMORY LOCATION. FOR EACH VALUE IN BITS <15:08>
4365                                  ;*      OF THE STACK POINTER ALL FOUR 'STACK BOUNDARY VALUES' ARE TESTED
4366                                  ;*      FOR THE CORRECT INDICATION.  MEMORY MANAGEMENT IS DISABLED FOR
4367                                  ;*      THIS TEST ONLY.  RELOCATION IS REENABLED FOR THE REST OF THE PROGRAM.
4368                                  ;*
4369                                  ;;*****************************************************************
4370  030454                          TST37:
4371  030454  000004                          SCOPE
4372  030456  012737  031562  001446          MOV     #TST40,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
4373                                                                  ;TEST FOR ESCAPE ON PARITY ERRORS
4374  030464  104410                          TBITO                   ;MAKE SURE T-BIT IS OFF FOR THIS TEST
4375  030466  042737  000001  177572  20$:     BIC     #BIT0,@#MMR0    ;DISABLE MEMORY MANAGEMENT FOR THIS TEST
```

```
4376  030474  005037  001202                CLR     $TMP2           ;INITIALIZE ERROR PRINT COUNTER
4377  030500  012702  000010                MOV     #10,R2          ;INITIALIZE STACK POINTER VALUE
4378  030504  012703  177400                MOV     #-400,R3        ;INITIALIZE STACK LIMIT REG. VALUE
4379  030510  062702  000400        1$:     ADD     #400,R2         ;LOAD NEW STACK POINTER VALUE IN R2
4380  030514  062703  000400                ADD     #400,R3         ;LOAD NEW STACK LIMIT REG VALUE IN R3
4381  030520  020227  030466                CMP     R2,#20$         ;IS SP VALUE LESS THAN LOC. OF THIS
4382                                                                  ;TEST CODE?
4383  030524  002404                        BLT     6$              ;BRANCH IF YES
4384  030526  020227  031530                CMP     R2,#10$         ;IS SP VALUE GREATER THAN LOC. OF THIS
4385                                                                  ;TEST CODE?
4386  030532  003001                        BGT     6$              ;BRANCH IF YES
4387  030534  000765                        BR      1$              ;OTHERWISE GO BACK AND INCREASE VALUE OF
4388                                                                  ;SP SO CODE IS NOT AFFECTED
4389  030536  005037  000000        6$:     CLR     @#0             ;CLEAR LOCATION 0 FOR RED ZONE CHECK
4390  030542  005037  001370                CLR     STKEXP          ;NO STACK VIOLATIONS EXPECTED(STKEXP=0)
4391  030546  012706  001100                MOV     #STACK,SP       ;INITIALIZE STACK POINTER
4392  030552  012737  031530  000004  2$:   MOV     #10$,ERRVEC     ;SET ERROR POINTER TO 10$
4393  030560  016204  177776                MOV     -2(R2),R4       ;SAVE STACK LOCATIONS TO BE USED
4394  030564  016205  177774                MOV     -4(R2),R5       ;EXIT TEST IF EITHER LOC. TIMES OUT
4395  030570  012737  030576  001110        MOV     #11$,$LPERR     ;SET LOOP ON ERROR POINTER TO 11$
4396  030576  012737  030762  000004  11$:   MOV     #50$,ERRVEC     ;SET ERROR POINTER TO 50$
4397  030604  010206                        MOV     R2,SP           ;LOAD STACK POINTER
4398  030606  010337  177774                MOV     R3,STKLMT       ;LOAD STACK LIMIT REGISTER
4399  030612  016666  177770  177770        MOV     -10(SP),-10(SP) ;REFERENCE LAST 'NO VIOLATION' ADDRESS
4400  030620  005037  000000                CLR     @#0             ;CLEAR LOCATION 0 FOR RED ZONE CHECK
4401  030624  012737  000001  001370        MOV     #1,STKEXP       ;EXPECT YELLOW ZONE VIOLATION(STKEXP=1)
4402  030632  012737  030640  001110        MOV     #12$,$LPERR     ;SET LOOP ON ERROR POINTER TO 12$
4403  030640  010206                12$:    MOV     R2,SP           ;LOAD STACK POINTER
4404  030642  016666  177766  177766        MOV     -12(SP),-12(SP) ;REFERENCE FIRST 'YELLOW ZONE' ADDRESS
4405  030650  000137  031442                JMP     60$             ;YELLOW ZONE VIOL. EXPECTED-GOT NONE
4406  030654  012737  030662  001110  3$:   MOV     #13$,$LPERR     ;SET LOOP ON ERROR POINTER TO 13$
4407  030662  010206                13$:    MOV     R2,SP           ;LOAD STACK POINTER
4408  030664  005037  000000                CLR     @#0             ;CLEAR LOCATION 0 FOR RED ZONE CHECK
4409  030670  016666  177730  177730        MOV     -50(SP),-50(SP) ;REFERENCE LAST 'YELLOW ZONE' ADDRESS
4410  030676  000137  031464                JMP     61$             ;YELLOW ZONE VIOL. EXPECTED-GOT NONE
4411  030702  012737  030710  001110  4$:   MOV     #14$,$LPERR     ;SET LOOP ON ERROR POINTER TO 14$
4412  030710  012737  000002  001370  14$:   MOV     #2,STKEXP       ;EXPECT RED ZONE VIOLATION(STKEXP=2)
4413  030716  010206                        MOV     R2,SP           ;LOAD STACK POINTER
4414  030720  005037  000000                CLR     @#0             ;CLEAR LOCATION 0 FOR RED ZONE CHECK
4415  030724  016237  177726  001200        MOV     -52(R2),$TMP1   ;SAVE CONTENTS OF TEST LOC.
4416  030732  005166  177726                COM     -52(SP)         ;REFERENCE FIRST 'RED ZONE' ADDRESS
4417  030736  000137  031506                JMP     62$             ;RED ZONE VIOL. EXPECTED-GOT NONE
4418  030742  013762  001200  177726  5$:   MOV     $TMP1,-52(R2)   ;RESTORE CONTENTS OF TEST LOC.
4419  030750  010462  177776                MOV     R4,-2(R2)       ;RESTORE STACK LOCATION USED
4420  030754  010562  177774                MOV     R5,-4(R2)       ;RESTORE STACK LOCATION USED
4421  030760  000653                        BR      1$              ;BRANCH UNTIL REACH NON EXSIST. MEM.
4422
4423
4424                                  ;
4425                                  ;
4426                                  ;THE CODE BELOW IS RESPONSIBLE FOR ALL OF
4427                                  ;THE ERROR REPORTING IN THIS TEST
4428                                  ;THE TEST VALUES OF THE STACK LIMIT REGISTER,STACK POINTER, AND
4429                                  ;THE STACK LOCATIONS TESTED ARE RESET TO THEIR NORMAL VALUES BUT
4430                                  ;THEIR TEST VALUES ARE RESTORED BEFORE RETURNING TO THE TEST.
4431                                  ;
```

```
4432
4433  030762  010637  001176          ;50$:  MOV     SP,$TMP0        ;SAVE TEST VALUE OF STACK POINTER
4434  030766  016200  177776                 MOV     -2(R2),R0       ;SAVE TEST VALUES IN TEST LOC.S
4435  030772  016201  177774                 MOV     -4(R2),R1
4436  030776  010462  177776                 MOV     R4,-2(R2)       ;RESTORE ORIGINAL VALUES TO
4437  031002  010562  177774                 MOV     R5,-4(R2)       ;       TEST LOCATIONS
4438  031006  005037  177774                 CLR     STKLMT          ;CLEAR STACK LIMIT REG. FOR ERROR REPORT
4439  031012  012706  001100                 MOV     #STACK,SP       ;RESET STACK POINTER FOR ERROR REPORTING
4440  031016  005737  001370                 TST     STKEXP          ;WAS A VIOLATION EXPECTED
4441  031022  001012                         BNE     51$             ;IF SO GO DECIDE WHAT KIND
4442  031024  104041                         ERROR   41              ;OTHERWISE ERROR - NO VIOLATION EXPECTED
4443  031026  013706  001176                 MOV     $TMP0,SP        ;RESTORE STACK POINTER VALUE
4444  031032  010062  177776                 MOV     R0,-2(R2)       ;RESTORE TEST VALUES TO TEST LOC.S
4445  031036  010162  177774                 MOV     R1,-4(R2)
4446  031042  010337  177774                 MOV     R3,STKLMT       ;RESTORE STACK LIMIT REG. TEST VALUE
4447  031046  000006                         RTT                     ;RETURN  FROM ERROR
4448
4449
4450
4451  031050  022737  000001  001370  51$:  CMP     #1,STKEXP       ;WAS A YELLOW ZONE VIOLATION EXPECTED
4452  031056  001076                         BNE     54$             ;IF NOT WE WERE EXPECTING A RED ZONE
4453  031060  005737  000000                 TST     @#0             ;WAS LOCATION 0 CHANGED
4454  031064  001401                         BEQ     52$             ;BRANCH IF IT WASN'T
4455  031066  104043                         ERROR   43              ;EXPECTING YELLOW ZONE - GOT RED
4456  031070  022737  000005  001202  52$:  CMP     #5,$TMP2        ;HAVE 5 "ERROR LOG" ERRORS BEEN TYPED?
4457  031076  003447                         BLE     53$             ;IF YES,DON'T TYPE ANY MORE
4458  031100  105737  001450                 TSTB    FORTY           ;EXECUTING ON AN 11/40?
4459  031104  001044                         BNE     53$             ;BRANCH AROUND LOG READS IF YES
4460                                         ;SEE IF YELLOW ZONE LOGGED CORRECTLY
4461  031106  010037  001204                 MOV     R0,$TMP3        ;SAVE THE CONTENTS OF R0
4462  031112  012737  000100  001206         MOV     #RLJAM,$TMP4    ;SAVE THE ADDRESS OF THE REG. IN CASE
4463                                         ;OF AN ERROR
4464  031120  012737  010000  001210         MOV     #BIT12,$TMP5    ;ALSO SAVE THE EXPECTED CONTENTS
4465  031126  076600                         MED                     ;READ THE CONTENTS OF THE ERROR LOG REG.
4466  031130  000100                         RLJAM                   ;SPECIFIED BY "RLJAM"
4467  031132  032700  010000                 BIT     #BIT12,R0       ;WAS THE RIGHT BIT SET
4468  031136  001003                         BNE     .+10            ;BRANCH IF IT WAS
4469  031140  005237  001202                 INC     $TMP2           ;INCREMENT ERROR PRINT COUNT
4470  031144  104022                         ERROR   22              ;ERROR LOG REG. DID NOT HAVE CORRECT BIT SET
4471  031146  013700  001204                 MOV     $TMP3,R0        ;RESTORE THE CONTENTS OF R0
4472  031152  010037  001204                 MOV     R0,$TMP3        ;SAVE THE CONTENTS OF R0
4473  031156  012737  000101  001206         MOV     #RLSERV,$TMP4   ;SAVE THE ADDRESS OF THE REG. IN CASE
4474                                         ;OF AN ERROR
4475  031164  012737  000001  001210         MOV     #BIT0,$TMP5     ;ALSO SAVE THE EXPECTED CONTENTS
4476  031172  076600                         MED                     ;READ THE CONTENTS OF THE ERROR LOG REG.
4477  031174  000101                         RLSERV                  ;SPECIFIED BY "RLSERV"
4478  031176  032700  000001                 BIT     #BIT0,R0        ;WAS THE RIGHT BIT SET
4479  031202  001003                         BNE     .+10            ;BRANCH IF IT WAS
4480  031204  005237  001202                 INC     $TMP2           ;INCREMENT ERROR PRINT COUNT
4481  031210  104022                         ERROR   22              ;ERROR LOG REG. DID NOT HAVE CORRECT BIT SET
4482  031212  013700  001204                 MOV     $TMP3,R0        ;RESTORE THE CONTENTS OF R0
4483  031216  005737  000000          53$:   TST     @#0             ;DID YOU GET RED ZONE?
4484  031222  001070                         BNE     57$             ;RETURN DIFFERENTLY IF YES
4485  031224  013706  001176                 MOV     $TMP0,SP        ;RESTORE STACK POINTER VALUE
4486  031230  010062  177776                 MOV     R0,-2(R2)       ;RESTORE TEST VALUES TO  TEST LOC.S
4487  031234  010162  177774                 MOV     R1,-4(R2)
```

```
4488  031240  010337  177774                MOV     R3,STKLMT       ;RESTORE STACK LIMIT REG. TEST VALUE
4489  031244  062762  000004  177774        ADD     #4,-4(R2)       ;FUDGE RETURN FROM TRAP
4490  031252  000006                        RTT                     ;RETURN FROM TRAP
4491
4492
4493
4494  031254  023762  001200  177726  54$:  CMP     $TMP1,-52(R2)   ;WAS INSTRUCTION ABORTED ON 'RED' VIOL.
4495  031262  001401                        BEQ     55$             ;BRANCH IF IT WAS EXECUTED
4496  031264  104045                        ERROR   45              ;RED ZONE DID NOT CAUSE ABORT
4497  031266  005737  001176          55$:  TST     $TMP0           ;IS TEST VALUE OF SP EQUAL TO LOC. 0
4498  031272  001401                        BEQ     56$             ;STACK SHOULD BE BUILT AT 0 ON A 'RED'
4499  031274  104046                        ERROR   46              ;EXPECTED RED ZONE - GOT YELLOW
4500  031276  022737  000005  001202  56$:  CMP     #5,$TMP2        ;HAVE 5 "ERROR LOG" ERRORS BEEN TYPED?
4501  031304  003437                        BLE     57$             ;IF YES, DON'T TYPE ANY MORE
4502  031306  105737  001450                TSTB    FORTY           ;EXECUTING ON AN 11/40?
4503  031312  001034                        BNE     57$             ;BRANCH AROUND LOG READS IF YES
4504                                                                ;SEE IF RED ZONE LOGGED CORRECTLY
4505  031314  010037  001204                MOV     R0,$TMP3        ;SAVE THE CONTENTS OF R0
4506  031320  012737  000100  001206        MOV     #RLJAM,$TMP4    ;SAVE THE ADDRESS OF THE REG. IN CASE
4507                                                                ;OF AN ERROR
4508  031326  012737  004010  001210        MOV     #004010,$TMP5   ;ALSO SAVE THE EXPECTED CONTENTS
4509  031334  076600                        MED                     ;READ THE CONTENTS OF THE ERROR LOG REG.
4510  031336  000100                        RLJAM                   ;SPECIFIED BY "RLJAM"
4511  031340  032700  004010                BIT     #004010,R0      ;WAS THE RIGHT BIT SET
4512  031344  001003                        BNE     .+10            ;BRANCH IF IT WAS
4513  031346  005237  001202                INC     $TMP2           ;INCREMENT ERROR PRINT COUNT
4514  031352  104022                        ERROR   22              ;ERROR LOG REG. DID NOT HAVE CORRECT BIT SET
4515  031354  013700  001204                MOV     $TMP3,R0        ;RESTORE THE CONTENTS OF R0
4516  031360  017737  147770  001410        MOV     @CPUERR,PCPUER  ;READ THE CPU ERROR REGISTER
4517  031366  032737  000004  001410        BIT     #BIT2,PCPUER    ;LOOK AT CONTENTS OF CPU ERROR REG.
4518  031374  001003                        BNE     57$             ;WAS RED ZONE VIOLATION REPORTED
4519  031376  005237  001202                INC     $TMP2           ;INCREMENT ERROR TYPE COUNT
4520  031402  104076                        ERROR   76              ;CPU ERROR REG.DID NOT REPORT RED ZONE
4521  031404  005737  000000          57$:  TST     @#0             ;DID WE GET RED ZONE?
4522  031410  001702                        BEQ     53$             ;RETURN DIFFERENTLY IF NO
4523  031412  013706  001176                MOV     $TMP0,SP        ;RESTORE STACK POINTER VALUE
4524  031416  010062  177776                MOV     R0,-2(R2)       ;RESTORE TEST VALUES TO TEST LOC.S
4525  031422  010162  177774                MOV     R1,-4(R2)
4526  031426  010337  177774                MOV     R3,STKLMT       ;RESTORE STACK LIMIT REG. TEST VALUE
4527  031432  062737  000004  000000        ADD     #4,@#0          ;FUDGE RETURN FROM TRAP
4528  031440  000006                        RTT                     ;RETURN FROM TRAP
4529                                ;
4530                                ;THIS GROUP OF CODING IS ENTERED WHEN A STACK VIOLATION
4531                                ;WAS EXPECTED BUT NONE OCCURRED
4532                                ;
4533  031442  005037  177774          60$:  CLR     STKLMT          ;CLEAR STACK LIMIT REG. FOR ERROR REPORT
4534  031446  012706  001100                MOV     #STACK,SP       ;RESET STACK POINTER FOR ERROR REPORT
4535  031452  104042                        ERROR   42              ;YELLOW ZONE EXPECTED - GOT NONE
4536  031454  010337  177774                MOV     R3,STKLMT       ;RESTORE STACK LIMIT REG. TEST VALUE
4537  031460  000137  030654                JMP     3$              ;RETRUN TO TEST AT 3$
4538  031464  005037  177774          61$:  CLR     STKLMT          ;CLEAR STACK LIMIT REG. FOR ERROR REPORT
4539  031470  012706  001100                MOV     #STACK,SP       ;RESET STACK POINTER FOR ERROR REPORT
4540  031474  104042                        ERROR   42              ;YELLOW ZONE EXPECTED - GOT NONE
4541  031476  010337  177774                MOV     R3,STKLMT       ;RESTORE STACK LIMIT REG. TEST VALUE
4542  031502  000137  030702                JMP     4$              ;RETURN TO TEST AT 4$
4543  031506  005037  177774          62$:  CLR     STKLMT          ;CLEAR STACK LIMIT REG. FOR ERROR REPORT
```

```
4544  031512  012706  001100                    MOV     #STACK,SP       ;RESET STACK POINTER FOR ERROR REPORT
4545  031516  104044                            ERROR   44              ;RED ZONE EXPECTED - GOT NONE
4546  031520  010337  177774                    MOV     R3,STKLMT       ;RESTORE STACK LIMIT REG. TEST VALUE
4547  031524  000137  030742                    JMP     5$              ;RETURN TO TEST AT 5$
4548                                    ;
4549                                    ;ONCE A NONEXSISTENT MEMORY LOCATION IS REACHED,
4550                                    ;THE STACK POINTER IS RESET, MEMORY MANAGEMENT IS
4551                                    ;REENABLED, AND THE TEST IS OVER
4552                                    ;
4553  031530  012737  015162  000004    10$:    MOV     #CPUER,ERRVEC   ;RESTORE NORMAL CPU TRAP ROUTINE
4554                                                                    ;GOT TIMEOUT, TEST IS OVER
4555  031536  012706  001100                    MOV     #STACK,SP       ;RESET THE STACK POINTER
4556  031542  052737  000001  177572            BIS     #BIT0,@#MMR0    ;REENABLE MEMORY MANAGEMENT
4557  031550  005037  177774                    CLR     @#STKLMT        ;LEAVE STACK LIMIT REG. CLEAR
4558  031554  012737  030466  001110            MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
4559  031562                            70$:
4560
4561
4562
4563
4564
4565                                    ;;*****************************************************************
4566                                    ;*TEST 40       MEMORY MANAGEMENT REGISTERS ONLY CLOCKED ONCE IF MMR0 NOT CLEARED
4567                                    ;*
4568                                    ;*      AS LONG AS THE 'ABORT' OR 'LOCKUP' SIGNAL IS ASSERTED (THAT IS AFTER
4569                                    ;*      AN ABORT AND UNTIL MMR0 BITS <15:13> ARE CLEARED) MMR0,
4570                                    ;*      AND MMR2 SHOULD NOT BE CLOCKED.  THIS TEST CAUSES A NON-RESIDENT
4571                                    ;*      ABORT, SAVES THE STATUS REGISTERS, CHANGES THE VECTOR TO 10$,
4572                                    ;*      AND THEN CAUSES A PAGE LENGTH ABORT.  AT THE SECOND ABORT THE
4573                                    ;*      STATUS REGISTERS ARE COMPARED WITH THEIR FIRST CONDITIONS. IF ANY
4574                                    ;*      OF THEM CHANGE, THE OLD AND THE NEW CONDITIONS WILL BE REPORTED.
4575                                    ;*
4576                                    ;;*****************************************************************
4577  031562                            TST40:
4578  031562  000004                            SCOPE
4579  031564  012737  032002  001446            MOV     #TST41,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
4580                                                                    ;TEST FOR ESCAPE ON PARITY ERRORS
4581  031572  104411                            TBITR                   ;RESTORE T-BIT IF IT WAS ON BEFORE
4582                                                                    ;THE LAST TEST
4583  031574  012737  031602  001106            MOV     #20$,$LPADR     ;SET LOOP ADDRESS POINTER TO 20$
4584                                                                    ;FOR ITERATION PASS
4585  031602  012737  000000  172310  20$:     MOV     #000000,KIPDR4  ;MAP PAGE 4 NON-RESIDENT, AND PAGE
4586                                                                    ;LENGTH OF 1 BLOCK
4587  031610  012737  031650  000250            MOV     #5$,MMVEC       ;SET M.M. TRAP VECTOR TO 5$
4588  031616  012737  000340  000252            MOV     #340,MMVEC+2    ;SET PRIORITY TO 7 GOTO KERNEL MODE
4589  031624  012737  031632  001110            MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
4590  031632  013700  100000            1$:     MOV     @#100000,R0     ;TRY TO READ THRU PAGE 4
4591                                                                    ;THIS PAGE NON-RESIDENT SHOULD CAUSE
4592                                                                    ;ABORT AND TRAP TO 5$.
4593  031636  012737  031672  000250  2$:      MOV     #10$,MMVEC      ;SET M.M. TRAP VECTOR TO 10$
4594  031644  013700  100100                    MOV     @#100100,R0     ;TRY TO READ FROM BLOCK 2 OF PAGE 4
4595                                                                    ;THIS SHOULD ABORT AGAIN BUT NONE
4596                                                                    ;OF THE MEMORY MANAGEMENT STATUS
4597                                                                    ;REGISTERS SHOULD BE CLOCKED THIS TIME.
4598
4599
```

```
4600  031650  013737  177572  001404  5$:     MOV     MMR0,PMMR0      ;READ MEMORY MANAGEMENT REGISTER 0
4601  031656  013737  177576  001406          MOV     MMR2,PMMR2      ;READ MEMORY MANAGEMENT REGISTER 2
4602  031664  012716  031636                  MOV     #2$,(KSP)       ;CHANGE RETURN ADDRESS TO 2$
4603  031670  000006                          RTT                     ;GO BACK TO 2$ AND CAUSE PAGE FAULT
4604
4605
4606  031672  012716  031700          10$:    MOV     #16$,(KSP)      ;CHANGE RETURN ADDRESS TO 16$
4607  031676  000006                          RTT                     ;RETURN TO 16$ AND CONTINUE PROGRAM
4608  031700  005037  001204          16$:    CLR     $TMP3           ;ERROR COUNTER, 3 POSSIBLE CMP FAILURES
4609  031704  013737  177572  001176          MOV     MMR0,$TMP0      ;READ MEMORY MANAGEMENT REGISTER 0
4610  031712  013737  177576  001202          MOV     MMR2,$TMP2      ;READ MEMORY MANAGEMENT REGISTER 2
4611  031720  023737  001202  001406          CMP     $TMP2,PMMR2     ;SEE IF MMR2 CHANGED
4612  031726  001402                          BEQ     12$             ;BRANCH IF MMR2 DIDN'T CHANGE
4613  031730  005237  001204                  INC     $TMP3           ;ONE COMPARE FAILURE
4614  031734  023737  001176  001404  12$:    CMP     $TMP0,PMMR0     ;SEE IF MMR0 CHANGED
4615  031742  001402                          BEQ     13$             ;BRANCH IF MMR0 DIDN'T CHANGE
4616  031744  005237  001204                  INC     $TMP3           ;ANOTHER COMPARE FAILURE
4617  031750  005737  001204          13$:    TST     $TMP3           ;WERE THERE ANY ERRORS ON THIS TEST
4618  031754  001401                          BEQ     19$             ;BRANCH IF NO ERRORS
4619  031756  104050                          ERROR   50              ;AT LEAST ONE M.M. REG CHANGED
4620  031760  042737  160556  177572  19$:    BIC     #160556,MMR0    ;CLEAR ALL ERROR BITS IN MMR0
4621  031766  012737  015432  000250          MOV     #MMTRAP,MMVEC   ;PUT BACK REGULAR M.M. TRAP ROUTINE
4622  031774  012737  031602  001110          MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
```

```
                                ;;*****************************************************************
                                ;*TEST 41       ERROR-ON-ERROR, KT ABORT & RED ZONE
                                ;*      THIS TEST CREATES AN "ERROR-ON-ERROR" SITUATION BY CAUSING
                                ;*      A KT ABORT (READ-ONLY VIOL.) IN USER  MODE FOLLOWED BY
                                ;*      ODD ADDRESS VIOL. BY USER STK. PTR.  THE PSW PICKED UP FROM
                                ;*      "MMVEC+2" PUTS IT IN USER MODE.  THEN WHEN TRYING TO
                                ;*      PUSH THE OLD PSW AND PC ON THE USER STACK, AN ODD ADDR.
                                ;*      VIOLATION OCCURS (KSP=1100, STACK LIMIT REG. = 703).
                                ;*      THE KERNEL STACK POINTER SHOULD BE FORCED TO THE VALUE 4,
                                ;*      BUILDING A STACK AT LOC.S 0&2 ON THE 11/6X.  ON THE 11/40
                                ;*      THE USER STACK POINTER IS FORCED TO 4, BUT THE KERNEL STACK
                                ;*      POINTER IS USED TO BUILD THE STACK AT LOCATIONS 1074 &
                                ;*      1076.  THE PSW THAT WILL BE SAVED ON THE STACK
                                ;*      SHOULD BE THE ONE PRESENT AT THE TIME OF THE SECOND ERROR
                                ;*      (THE ONE FROM "MMVEC+2") FOR THE 11/6X, OR THE PSW PRESENT
                                ;*      AT THE TIME OF THE FIRST ERROR FOR THE 11/40
                                ;*          THE PC SAVED ON THE STACK, IN BOTH CASES
                                ;*      SHOULD BE THE UPDATED PC FROM THE FIRST ERROR (THE KT
                                ;*      ABORT INSTRUCTION).
                                ;*          ALSO, MMR0 SHOULD RECORD A R/O VIOL. & USER PAGE 0,
                                ;*      MMR2 SHOULD LOCK UP THE PC OF THE KT ABORT INSTRUCTION,
                                ;*      AND THE WHAMI AND LOG WHAMI REGISTERS SHOULD RECORD AN
                                ;*      "ERROR-ON-ERROR" ON THE 11/6X.
                                ;*
                                ;;*****************************************************************
032002                          TST41:
032002  000004                          SCOPE
032004  012737  032640  001446          MOV     #TST42,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
                                                                ;TEST FOR ESCAPE ON PARITY ERRORS
                                        .EQUIV  BIT4,TBIT       ;BIT 4 OF P.S. IS T-BIT TRAPPING BIT
032012  104410                          TBITO                   ;MAKE SURE T-BIT IS OFF FOR THE NEXT 3 TESTS
032014  012737  000340  000006  20$:    MOV     #340,ERRVEC+2   ;SET PRIORITY OF ERRVEC TO 7
                                                                ;AND MAKE NEW MODE KERNEL
032022  012737  000000  177640          MOV     #000,UIPAR0     ;MAP USER PAGE 0 TO PHYS. 0
032030  012737  000200  177642          MOV     #200,UIPAR1     ;MAP USER PAGE 1 TO 4K -8K
032036  012737  000400  177644          MOV     #400,UIPAR2     ;MAP USER PAGE 2 TO 8K - 12K
032044  012737  000600  177646          MOV     #600,UIPAR3     ;MAP USER PAGE 3 TO 12K - 16K
032052  012737  007600  177656          MOV     #7600,UIPAR7    ;MAP USER PAGE 7 TO I/O PAGE
032060  012737  077406  177600          MOV     #77406,UIPDR0   ;MAKE USER PAGE 0 200 BLCKS, R/W
032066  012737  077406  177602          MOV     #77406,UIPDR1   ;MAKE USER PAGE 1 200 BLCKS, R/W
032074  012737  077406  177604          MOV     #77406,UIPDR2   ;MAKE USER PAGE 2 200 BLCKS, R/W
032102  012737  077406  177606          MOV     #77406,UIPDR3   ;MAKE USER PAGE 3 200 BLCKS, R/W
032110  012737  077406  177616          MOV     #77406,UIPDR7   ;MAKE USER PAGE 7 200 BLOCKS, R/W
032116  012737  032166  001110          MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
032124  012737  032172  000004          MOV     #10$,ERRVEC     ;SET ERRVEC TO 10$
032132  012737  140017  000252          MOV     #140017,MMVEC+2 ;LOAD NEW PSW INTO MMVEC+2
032140  012737  000600  177650          MOV     #600,UIPAR4     ;MAP USER PAGE 4 TO 12K
032146  012737  077402  177610          MOV     #77402,UIPDR4   ;MAKE USER PAGE 4 200 BLOCKS, READ-ONLY
032154  012737  140000  177776          MOV     #140000,PSW     ;GO TO USER MODE
032162  012706  000703                  MOV     #703,USP        ;SETUP USER STACK POINTER
                                                                ;TO CAUSE ODD ADDR. ERROR
032166  005237  100000          1$:     INC     @#100000        ;CREATE KT ABORT (R/O VIOL.) WHICH
                                                                ;PICKS UP A USER PSW AND THEN
                                                                ;CAUSES A ODD ADDR. VIOLATION.
                                                                ;THE DOUBLE ERROR SHOULD BE PROCESSED
                                                                ;AND THEN THE PROGRAM SHOULD CONTINUE
```

```
4679
4680  032172  016602  000002          10$:    MOV     2(KSP),R2       ;AT 10$ POINTED TO BY "ERRVEC"
4681  032176  011603                          MOV     (KSP),R3        ;SAVE PSW IN R2
4682  032200  012737  077406  177610          MOV     #77406,UIPDR4   ;SAVE PC IN R3
4683  032206  010601                          MOV     KSP,R1          ;MAKE USER PAGE 4 READ & WRITE
4684  032210  012711  032224                  MOV     #16$,(R1)       ;SAVE THE KERNEL STACK POINTER
4685  032214  012761  000340  000002          MOV     #340,2(R1)      ;SET RETURN PC TO 16$
4686  032222  000006                          RTT                     ;SET RETURN PSW TO KERNEL MODE
4687  032224  042761  000020  000002  16$:    BIC     #TBIT,2(R1)     ;RETURN FROM "ERR-ON-ERR" SERVICE
4688  032232  012706  001100                  MOV     #STACK,KSP      ;CLEAR T BIT IN SAVED PSW
4689  032236  005037  001200                  CLR     $TMP1           ;RESTORE THE STACK POINTER
4690  032242  012737  140017  001202          MOV     #140017,$TMP2   ;THIS IS THE ERROR COUNTER FLAG
4691                                                                  ;LOAD EXPECTED PS SAVED FOR 11/6X
4692  032250  105737  001450                  TSTB    FORTY           ;(PS FROM MMVEC+2 FETCHED BY KT ABORT)
4693  032254  001403                          BEQ     19$             ;EXECUTING ON AN 11/40?
4694  032256  012737  140000  001202          MOV     #140000,$TMP2   ;BRANCH IF NO
4695                                                                  ;LOAD EXPECTED PS SAVED FOR 11/40
4696  032264  023702  001202          19$:    CMP     $TMP2,R2        ;(PS AT TIME OF FIRST (KT) ERROR)
4697  032270  001402                          BEQ     18$             ;WAS CORRECT PS SAVED?
4698  032272  005237  001200                  INC     $TMP1           ;BRANCH IF PS IS CORRECT
4699  032276  022703  032172          18$:    CMP     #10$,R3         ;RIGHT PS IS ON STACK
4700                                                                  ;THE PC THAT SHOULD HAVE BEEN
4701                                                                  ;ON THE STACK IS THE UPDATED
4702                                                                  ;PC FROM THE KT ABORT INST.
4703  032302  001402                          BEQ     11$             ;(10$ = 1$+4)
4704  032304  005237  001200                  INC     $TMP1           ;BRANCH IF PC CORRECT
4705  032310  013737  177572  001404  11$:    MOV     MMR0,PMMR0      ;WRONG PC ON THE STACK
4706  032316  013737  177576  001406          MOV     MMR2,PMMR2      ;SAVE MMR0 FOR COMPARE
4707  032324  105737  001450                  TSTB    FORTY           ;SAVE MMR2, IT SHOULD EQUAL #10
4708  032330  001040                          BNE     21$             ;EXECUTING ON AN 11/40?
4709  032332  010037  001204                  MOV     R0,$TMP3        ;BRANCH IF YES
4710  032336  076600                          MED                     ;SAVE CONTENTS OF R0
4711  032340  000105                          RLWHAM                  ;READ LOG WHAMI REG.
4712  032342  030027  000004                  BIT     R0,#BIT2        ;WAS ERROR-ON-ERROR BIT SET?
4713  032346  001007                          BNE     17$             ;BRANCH IF YES
4714  032350  012737  000004  001210          MOV     #BIT2,$TMP5     ;SAVE EXPECTED CONTENTS FOR ERROR
4715  032356  012737  000105  001206          MOV     #RLWHAM,$TMP4   ;SAVE ADDRESS FOR ERROR
4716  032364  104022                          ERROR   22              ;ERR-ON-ERR BIT NOT SET IN
4717  032366                          17$:                            ;LOG WHAMI REG.
4718  032366  012737  000022  001206          MOV     #RWHAM,$TMP4    ;SAVE ADDR. FOR ERROR
4719  032374  012737  000004  001210          MOV     #BIT2,$TMP5     ;SAVE EXP. DATA FOR ERROR
4720  032402  076600                          MED                     ;MED READ THE WHAMI REG.
4721  032404  000022                          RWHAM
4722  032406  032700  000004                  BIT     #BIT2,R0        ;WAS "ERR-ON-ERR" BIT SET?
4723  032412  001001                          BNE     .+4             ;BRANCH IF YES
4724  032414  104022                          ERROR   22              ;"ERR-ON-ERR" BIT NOT SET
4725  032416  042700  000004                  BIC     #BIT2,R0        ;CLEAR BIT2 IN R0
4726  032422  076600                          MED                     ;WRITE ORIGINAL CONTENTS
4727  032424  000222                          WWHAM                   ;TO WHAMI WITH BIT2=0
4728  032426  013700  001204                  MOV     $TMP3,R0        ;RESTORE CONTENTS OF R0
4729  032432  022737  020151  001404  21$:    CMP     #020151,PMMR0   ;SHOULD HAVE READ-ONLY VIO.
4730                                                                  ;COMPLETED, PAGE 4, USER I-SPACE.
4731  032440  001402                          BEQ     12$             ;BRANCH IF CORRECT CONDITION
4732  032442  005237  001200                  INC     $TMP1           ;WRONG ABORT CONDITION
4733  032446  105737  001450          12$:    TSTB    FORTY           ;EXECUTING ON AN 11/40?
4734  032452  001424                          BEQ     23$             ;BRANCH IF NO AROUND 11/40 CHECKS
```

```
4735  032454  052737  140000  177776          BIS     #140000,PSW     ;GO TO USER MODE
4736  032462  010604                          MOV     USP,R4          ;READ THE USER STACK POINTER
4737  032464  042737  140000  177776          BIC     #140000,PSW     ;GO BACK TO KERNEL MODE
4738  032472  022704  000004                  CMP     #4,R4           ;WAS USER STK. PTR. FORCED TO 4 ON 11/40?
4739  032476  001404                          BEQ     22$             ;BRANCH IF YES
4740  032500  005237  001200                  INC     $TMP1           ;USP ON 11/40 IS NOT CORRECT
4741  032504  010401                          MOV     R4,R1           ;SAVE VALUE OF USP FOR TYPEOUT
4742  032506  000413                          BR      13$             ;DON'T BOTHER CHECKING KSP IF USP WAS WRONG
4743  032510  022701  001074          22$:    CMP     #KERSTK-4,R1    ;WAS PC & PSW SAVED ON 11/40 KERNEL STACK?
4744  032514  001410                          BEQ     13$             ;BRANCH IF YES
4745  032516  005237  001200                  INC     $TMP1           ;PC & PSW NOT SAVED ON 11/40 KERNEL STACK
4746  032522  000405                          BR      13$             ;BRANCH AROUND 11/6X KSP CHECK
4747  032524                          23$:
4748  032524  022701  000000                  CMP     #0,R1           ;THE KER. STK. POINTER
4749                                                                  ;SHOULD HAVE BEEN = 0
4750                                                                  ;AFTER ERR-ON-ERR TRAP TO LOC. 4
4751  032530  001402                          BEQ     13$             ;BRANCH IF KSP IS CORRECT
4752  032532  005237  001200                  INC     $TMP1           ;KSP IS NOT CORRECT
4753  032536  022737  032166  001406  13$:    CMP     #1$,PMMR2       ;SEE IF MMR2 EQUALS ADDRS. OF KT ABORT
4754  032544  001402                          BEQ     14$             ;BRANCH IF ADDRESS IS CORRECT
4755  032546  005237  001200                  INC     $TMP1           ;MMR2 HAS THE WRONG ADDRESS
4756  032552  005737  001200          14$:    TST     $TMP1           ;DID ANY OF THE COMPARES FAIL?
4757  032556  001401                          BEQ     15$             ;BRANCH IF ALL COMPARES SUCCEEDED
4758  032560  104047                          ERROR   47              ;AT LEAST ONE COMPARE FAILED
4759                                                                  ;IF USER STK. PTR. WRONG ON 11/40,
4760                                                                  ;THE KERNEL STK. PTR. WAS NOT CHECKED
4761  032562  052737  000340  177776  15$:    BIS     #340,PSW        ;NORMAL PRIORITY IS 7
4762  032570  052737  140000  177776          BIS     #140000,PSW     ;GO TO USER MODE TO RESET PTR
4763  032576  012706  000700                  MOV     #700,USP        ;RESTORE USER STK PTR TO 700
4764  032602  042737  140000  177776          BIC     #140000,PSW     ;RETURN TO KERNEL MODE
4765  032610  012737  000340  000006          MOV     #340,ERRVEC+2   ;RESTORE CORRECT PS TO ERROR VECTOR
4766  032616  012737  032014  001110          MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
4767  032624  042737  160556  177572          BIC     #160556,MMR0    ;CLEAR ERROR CONDITION IN MMR0
4768  032632  012737  015162  000004          MOV     #CPUER,ERRVEC   ;RESTORE NORMAL CPU TRAP ROUTINE
```

```
                        ;;**********************************************************************
                        ;*TEST 42       USER MODE, ABORT VECTOR FROM KERNEL SPACE
                        ;*
                        ;*      THIS TEST DOES AN ABORT FROM USER MODE.  THE VECTOR
                        ;*      SHOULD BE PICKED UP FROM KERNEL SPACE
                        ;*      FORCING KERNEL MODE DURING THE ABORT SEQUENCE.
                        ;*
                        ;*      THE 'HALTS' IN THIS TEST ARE SPACE FILLERS AND SHOULD NEVER BE
                        ;*      REACHED, IF USER MODE IS ENABLED PROPERLY
                        ;*
                        ;*
                        ;*
                        ;;**********************************************************************
032640                  TST42:
032640  000004                  SCOPE
032642  012737  033156  001446          MOV     #TST43,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
                                                                ;TEST FOR ESCAPE ON PARITY ERRORS
032650  005037  001366  20$:     CLR     MMEXP                   ;NOT EXPECTING ANY M.M. TRAPS YET
032654  012700  077406          MOV     #77406,R0               ;PAGE LENGTH-200 BLOCKS EXPAND UP
                                                                ;RESIDENT READ/WRITE
032660  010037  177600          MOV     R0,UIPDR0               ;LOAD USER PAGE 0
032664  010037  177602          MOV     R0,UIPDR1               ;LOAD USER PAGE 1
032670  010037  177604          MOV     R0,UIPDR2               ;LOAD USER PAGE 2
032674  010037  177606          MOV     R0,UIPDR3               ;LOAD USER PAGE 3
032700  010037  177616          MOV     R0,UIPDR7               ;LOAD USER PAGE 7
032704  012737  000001  177640  MOV     #1,UIPAR0               ;MAP USER PAGE 0 TO 00100
032712  012737  000201  177642  MOV     #201,UIPAR1             ;MAP USER PAGE 1 TO 20100
032720  012737  000401  177644  MOV     #401,UIPAR2             ;MAP USER PAGE 2 TO 40100
032726  012737  000601  177646  MOV     #601,UIPAR3             ;MAP USER PAGE 3 TO 60100
032734  012737  007600  177656  MOV     #7600,UIPAR7            ;MAP USER PAGE 7 TO I/O PAGE
032742  012737  077400  177612  MOV     #77400,UIPDR5           ;MAKE USER PAGE 5 NON-RESIDENT
032750  012737  000600  177652  MOV     #600,UIPAR5             ;MAP USER PAGE 5 TO 12K
032756  012737  015574  000350  MOV     #USEVEC,350             ;USER SPACE VECTOR
032764  012737  000000  000352  MOV     #000,352                ;USER SPACE PSW = 000
032772  012737  033000  001110  MOV     #5$,$LPERR              ;SET LOOP ON ERROR POINTER TO 5$
033000  012737  140000  177776  5$:     MOV     #140000,PSW     ;GO TO USER MODE.
                                                                ;THE NEXT INSTRUCTION EXECUTED IS AT
                                                                ;3$.  THE ADDRESS IS 100 OCTAL BYTES
                                                                ;GREATER THAN THE ADDRESS AT 1$.
033006  104051          1$:     ERROR   51                      ;DIDN'T GO TO USER MODE
033010  005037  177776          CLR     PSW                     ;GO BACK INTO KERNEL MODE
033014  000137  033056          JMP     2$                      ;GO TO EXIT OF TEST
033020  000000                  HALT                            ;THE NEXT 'SEVERAL' HALTS SHOULDN'T
033022  000000                  HALT                            ;EVER BE REACHED
033024  000000                  HALT                            ;EVER BE REACHED
033026  000000                  HALT                            ;EVER BE REACHED
033030  000000                  HALT                            ;EVER BE REACHED
033032  000000                  HALT                            ;EVER BE REACHED
033034  000000                  HALT                            ;EVER BE REACHED
033036  000000                  HALT                            ;EVER BE REACHED
033040  000000                  HALT                            ;EVER BE REACHED
033042  000000                  HALT                            ;EVER BE REACHED
033044  000000                  HALT                            ;EVER BE REACHED
033046  000000                  HALT                            ;EVER BE REACHED
```

```
4825  033050  000000                          HALT                    ;EVER BE REACHED
4826  033052  000000                          HALT                    ;EVER BE REACHED
4827  033054  000000                          HALT                    ;EVER BE REACHED
4828  033056  012737  015432  000250  2$:     MOV     #MMTRAP,MMVEC   ;RESTORE NORMAL M.M. TRAP ROUTINE
4829  033064  022737  100153  001404          CMP     #100153,PMMR0                   ;EXPECTING NON-RESIDENT PAGE 5
4830                                                                                  ;ABORT IN USER MODE I-SPACE
4831  033072  001401                          BEQ     11$                             ;BRANCH IF CORRECT COND
4832  033074  104052                          ERROR   52                              ;ABORT CONDITION INCORRECT
4833  033076  012737  032650  001110  11$:    MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
4834  033104  000424                          BR      TST43           ;;BRANCH TO NEXT TEST
4835  033106  012737  015544  000150  3$:     MOV     #KERVEC,<MMVEC-100>     ;SET UP KERNEL SPACE VECTOR
4836  033114  012737  000340  000152          MOV     #340,<MMVEC+2-100>      ;KERNEL SPACE PSW = 340
4837  033122  013700  120000                  MOV     @#120000,R0             ;READ FROM PAGE 5, USER SPACE
4838  033126  000405                          BR      10$                     ;GO SEE IF GOT EXPECTED ABORT
4839                                                                          ;AFTER WE'VE JUST RETURNED
4840                                                                          ;FROM "KERVEC"
4841                                                                          ;THESE HALTS SHOULDN'T
4842  033130  000000                          HALT                            ;EVER BE REACHED
4843  033132  000000                          HALT
4844  033134  000000                          HALT
4845  033136  000000                          HALT
4846  033140  000000                          HALT
4847  033142  042737  160556  177572  10$:    BIC     #160556,MMR0                    ;CLEAR MMR0 FOR NEXT TEST
4848  033150  012737  000340  177776          MOV     #340,PSW        ;GO BACK INTO KERNEL MODE NOW
4849                                          ;THE NEXT INSTRUCTION TO BE EXECUTED
4850                                          ;IS AT LABEL 2$
4851
4852
4853                          ;;*****************************************************************
4854                          ;*TEST 43       SECOND BIT TEST OF MEMORY MANAGEMENT REGISTER 2
4855                          ;*
4856                          ;*      THIS TEST MAPS USER PAGES 3,4,5, AND 6 TO 12K READ-ONLY.
4857                          ;*      A VIRTUAL ADDRESS IS THEN FORMED IN REGISTER 1 USING BITS
4858                          ;*      <15:13> TO SELECT PAR/PDRS 3 THRU 6 AND A "1" WHICH IS SHIFTED
4859                          ;*      THRU BITS <12:1>.  THE INSTRUCTION "INC (R1)" (CODED AS 005211)
4860                          ;*      IS THEN PLACED AT THE VIRTUAL ADDRESS IN R1 CAUSING A KT-
4861                          ;*      READ ONLY VIOLATION WHEN EXECUTED.  THE CONTENTS OF MMR2 ARE
4862                          ;*      TESTED ALONG WITH BITS <06:05> AND <03:01> OF MMR0.
4863                          ;*
4864                          ;;*****************************************************************
4865  033156                  TST43:
4866  033156  000004                  SCOPE
4867  033160  012737  033662  001446          MOV     #TST44,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
4868                                                                  ;TEST FOR ESCAPE ON PARITY ERRORS
4869  033166  012737  000002  001212          MOV     #2,$TIMES       ;;DO 2 ITERATIONS
4870  033174                          20$:
4871  033174  012737  077406  172300          MOV     #77406,KIPDR0   ;MAKE KERNEL I PAGE 0 200 BLOCKS, R/W
4872  033202  012737  077406  172302          MOV     #77406,KIPDR1   ;MAKE KERNEL I PAGE 1 200 BLOCKS, R/W
4873  033210  012737  077406  172304          MOV     #77406,KIPDR2   ;MAKE KERNEL I PAGE 2 200 BLOCKS, R/W
4874  033216  012737  077406  172306          MOV     #77406,KIPDR3   ;MAKE KERNEL I PAGE 3 200 BLOCKS, R/W
4875  033224  012737  077406  172316          MOV     #77406,KIPDR7   ;MAKE KERNEL I PAGE 7 200 BLOCKS, R/W
4876  033232  012737  000000  172340          MOV     #000,KIPAR0     ;MAP KERNEL I PAGE 0 TO 0 - 4K
4877  033240  012737  000200  172342          MOV     #200,KIPAR1     ;MAP KERNEL I PAGE 1 TO 4K - 8K
4878  033246  012737  000400  172344          MOV     #400,KIPAR2     ;MAP KERNEL I PAGE 2 TO 8K - 12K
4879  033254  012737  000600  172346          MOV     #600,KIPAR3     ;MAP KERNEL I PAGE 3 TO 12K - 16K
4880  033262  012737  007600  172356          MOV     #7600,KIPAR7    ;MAP KERNEL I PAGE 7 TO THE I/O PAGE
```

```
4881  033270  012737  000001  177572          MOV     #BIT0,MMR0      ;ENABLE 18-BIT RELOCATION IF NOT ON
4882  033276  012700  077406                  MOV     #77406,R0
4883  033302  010037  172310                  MOV     R0,KIPDR4       ;MAP KERNEL PAGE 4 READ-WRITE
4884  033306  010037  172312                  MOV     R0,KIPDR5       ;MAP KERNEL PAGE 5 READ-WRITE
4885  033312  010037  172314                  MOV     R0,KIPDR6       ;MAP KERNEL PAGE 6 READ-WRITE
4886  033316  012700  000600                  MOV     #600,R0
4887  033322  010037  172350                  MOV     R0,KIPAR4       ;MAP KERNEL PAGE 4 TO 12K
4888  033326  010037  172352                  MOV     R0,KIPAR5       ;MAP KERNEL PAGE 5 TO 12K
4889  033332  010037  172354                  MOV     R0,KIPAR6       ;MAP KERNEL PAGE 6 TO 12K
4890  033336  012737  077402  177606          MOV     #77402,UIPDR3   ;MAP USER PAGE 3 READ-ONLY
4891  033344  012737  077402  177610          MOV     #77402,UIPDR4   ;MAP USER PAGE 4 READ-ONLY
4892  033352  012737  077402  177612          MOV     #77402,UIPDR5   ;MAP USER PAGE 5 READ-ONLY
4893  033360  012737  077402  177614          MOV     #77402,UIPDR6   ;MAP USER PAGE 6 READ-ONLY
4894  033366  012737  000000  177640          MOV     #000,UIPAR0     ;MAP USER PAGE 0 TO 0-4K
4895  033374  012737  000200  177642          MOV     #200,UIPAR1     ;MAP USER PAGE 1 TO 4K
4896  033402  012737  000400  177644          MOV     #400,UIPAR2     ;MAP USER PAGE 2 TO 8K
4897  033410  012737  000600  177646          MOV     #600,UIPAR3     ;MAP USER PAGE 3 TO 12K
4898  033416  012737  000600  177650          MOV     #600,UIPAR4     ;MAP USER PAGE 4 TO 12K
4899  033424  012737  000600  177652          MOV     #600,UIPAR5     ;MAP USER PAGE 5 TO 12K
4900  033432  012737  000600  177654          MOV     #600,UIPAR6     ;MAP USER PAGE 6 TO 12K
4901  033440  012737  033476  001110          MOV     #3$,$LPERR      ;SET LOOP ON ERROR POINTER TO 3$
4902  033446  012700  140000                  MOV     #140000,R0      ;THIS WILL FORCE USER MODE WHEN USED
4903                                                                  ;AS A PROCESSOR STATUS
4904  033452  012704  060000                  MOV     #60000,R4       ;SETUP BITS (15:13) IN R4 SO V.A. WILL
4905                                                                  ;INITIALLY SELECT PAR/PDR 3
4906  033456  012737  033522  000250          MOV     #10$,MMVEC      ;SET M.M. TRAP VECTOR TO 10$
4907  033464  012737  000340  000252          MOV     #340,MMVEC+2    ;MAKE SURE TRAP TAKES YOU TO KERNEL
4908  033472  012701  000002          2$:     MOV     #2,R1           ;PUT TEST BIT AT STARTING POSTION(BIT 1)
4909  033476  042701  160000          3$:     BIC     #160000,R1      ;CLEAR BITS (15:13) SO APF SELECT BITS CAN
4910                                                                  ;BE RESET
4911  033502  050401                          BIS     R4,R1           ;SET BITS (15:13) TO SELECT PAR/PDR BEING USED
4912  033504  011137  001176                  MOV     (R1),$TMP0      ;SAVE CONTENTS OF TEST LOCATION
4913  033510  012711  005211                  MOV     #5211,(R1)      ;PUT "ABORT INSTRUCTION" AT TEST LOC.
4914  033514  010046                          MOV     R0,-(KSP)       ;PUSH USER PS ON STACK
4915  033516  010146                          MOV     R1,-(KSP)       ;PUSH USER'S VIRTUAL PC ON STACK
4916  033520  000002                          RTI                     ;RETURN TO USER MODE
4917                                                                  ;THE FIRST INSTRUCTION FETCH WILL
4918                                                                  ;CAUSE A NON-RESIDENT ABORT AND LOCK
4919                                                                  ;MMR2 SO THAT ALL BITS CAN BE CHECKED.
4920
4921
4922
4923  033522  062706  000004          10$:    ADD     #4,KSP          ;CLEAN UP STACK FOR NEXT TIME
4924  033526  013711  001176                  MOV     $TMP0,(R1)      ;RESTORE CONTENTS OF TEST LOCATION
4925  033532  013737  177576  001406          MOV     MMR2,PMMR2      ;READ MMR2 TO TEMP LOCATION
4926  033540  013737  177572  001404          MOV     MMR0,PMMR0      ;READ MMR0 TO TEMP LOCATION
4927  033546  020137  001406                  CMP     R1,PMMR2        ;SEE IF MMR2 LOCKED CORRECT V. A.
4928  033552  001401                          BEQ     11$             ;BRANCH IF MMR2 HAS RIGHT V.A.
4929  033554  104053                          ERROR   53              ;WRONG V.A.
4930  033556  005002                  11$:    CLR     R2              ;R2 WILL GET THE VIRTUAL PAGE NO.
4931  033560  010103                          MOV     R1,R3           ;COPY VIRTUAL ADDRESS INTO R3
4932  033562  012705  000003                  MOV     #3,R5           ;COMBINED LEFT SHIFT (R2,R3) 3 BITS
4933  033566  000241                  4$:     CLC
4934  033570  006303                          ASL     R3
4935  033572  006102                          ROL     R2
4936  033574  077504                          SOB     R5,4$
```

```
4937  033576  006102                          ROL     R2              ;ADJUST PAGE NUMBER
4938  033600  052702  020141                  BIS     #020141,R2      ;SET OTHER EXPECTED BITS IN MMR0
4939  033604  020237  001404                  CMP     R2,PMMR0        ;SEE IF MMR0 RECORDED CORRECT PAGE NO.
4940  033610  001401                          BEQ     12$             ;BRANCH IF PAGE NUMBER WAS CORRECT
4941  033612  104054                          ERROR   54              ;WRONG PAGE NO. IN MMR0
4942  033614  042737  160556  177572  12$:    BIC     #160556,MMR0    ;CLEAR ALL ERROR BITS IN MMR0
4943  033622  032701  010000                  BIT     #BIT12,R1       ;HAS BIT BEEN SHIFTED ALL THE WAY THRU
4944  033626  001002                          BNE     13$             ;BRANCH IF IT WAS
4945  033630  006301                          ASL     R1              ;SHIFT BIT ONE POSITION LEFT
4946  033632  000721                          BR      3$              ;GO AND TRY THIS VIRTUAL ADDRESS
4947  033634  062704  020000          13$:    ADD     #20000,R4       ;INCREASE R4 TO SELECT NEXT PAR/PDR
4948  033640  022704  160000                  CMP     #160000,R4      ;HAVE ALL PAGES(3,4,5,6) BEEN USED
4949  033644  001312                          BNE     2$              ;CONTINUE IF THEY HAVEN'T
4950  033646  012737  015432  000250          MOV     #MMTRAP,MMVEC   ;PUT BACK REGULAR M.M. TRAP ROUTINE
4951  033654  012737  033174  001110          MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
4952
4953
4954
4955
4956                                  .SBTTL  ************ ENTRY POINT 6 --- STARTING ADDRESS 224 ************
4957                                  .SBTTL  **********  W BIT LOGIC TEST AND DUAL MAPPING TESTS *********
4958                                  ;*
4959                                  ;*      THIS GROUP OF TESTS CHECKS OUT THE W-BIT LOGIC
4960                                  ;*      AND THEN USES THAT LOGIC TO VERIFY THAT THERE IS NO
4961                                  ;*      DUAL MAPPING OF PAR/PDR PAIRS BETWEEN GROUPS OR INSIDE A GROUP.
4962                                  ;*      ALL W-BITS ARE SET TO ENSURE THAT THERE ISN'T A BAD
4963                                  ;*      CHIP IN THE PDR'S.
4964                                  ;*
4965
4966
4967                                  ;;*********************************************************************
4968                                  ;*TEST 44       TEST W-BIT LOGIC
4969                                  ;*
4970                                  ;*      THIS TEST CHECKS ALL THE LOGIC FOR THE W-BIT ON
4971                                  ;*      'K504'.  THE BITS ARE SET ONE AT A TIME THEN A REFERENCE
4972                                  ;*      TO THAT PAGE CAUSING AN ABORT IS MADE TO SEE IF THE BIT REMAINS
4973                                  ;*      SET.  LAST EITHER THE PAR OR PDR IS WRITTEN TO SEE THAT THE
4974                                  ;*      BIT IS CLEARED DURING A PAR OR PDR LOAD.
4975                                  ;*
4976                                  ;;*********************************************************************
4977  033662                          TST44:
4978  033662  000004                          SCOPE
4979  033664  012737  034342  001446          MOV     #TST45,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
4980                                                                  ;TEST FOR ESCAPE ON PARITY ERRORS
4981  033672  104411                          TBITR                   ;RESTORE T-BIT IF IT WAS ON BEFORE THE
4982                                                                  ;LAST 3 TESTS
4983  033674  012737  034026  001106  ENTPT6: MOV     #20$,$LPADR     ;SET LOOP ADDRESS POINTER TO 20$
4984  033702  012737  034026  001110          MOV     #20$,$LPERR     ;SET LOOP ON ERROR POINTER TO 20$
4985  033710  012737  000044  001102          MOV     #44,$TSTNM      ;LOAD TEST NUMBER INTO MEMORY
4986  033716  013777  001102  145216          MOV     $TSTNM,@DISPLAY ;DISPLAY TEST NUMBER FOR THIS TEST
4987  033724  012737  077406  172300          MOV     #77406,KIPDR0   ;MAKE KERNEL I PAGE 0 200 BLOCKS, R/W
4988  033732  012737  077406  172302          MOV     #77406,KIPDR1   ;MAKE KERNEL I PAGE 1 200 BLOCKS, R/W
4989  033740  012737  077406  172304          MOV     #77406,KIPDR2   ;MAKE KERNEL I PAGE 2 200 BLOCKS, R/W
4990  033746  012737  077406  172306          MOV     #77406,KIPDR3   ;MAKE KERNEL I PAGE 3 200 BLOCKS, R/W
4991  033754  012737  077406  172316          MOV     #77406,KIPDR7   ;MAKE KERNEL I PAGE 7 200 BLOCKS, R/W
4992  033762  012737  000000  172340          MOV     #000,KIPAR0     ;MAP KERNEL I PAGE 0 TO 0 - 4K
```

```
 4993  033770  012737  000200  172342          MOV     #200,KIPAR1     ;MAP KERNEL I PAGE 1 TO 4K - 8K
 4994  033776  012737  000400  172344          MOV     #400,KIPAR2     ;MAP KERNEL I PAGE 2 TO 8K - 12K
 4995  034004  012737  000600  172346          MOV     #600,KIPAR3     ;MAP KERNEL I PAGE 3 TO 12K - 16K
 4996  034012  012737  007600  172356          MOV     #7600,KIPAR7    ;MAP KERNEL I PAGE 7 TO THE I/O PAGE
 4997  034020  012737  000001  177572          MOV     #BIT0,MMR0      ;ENABLE 18-BIT RELOCATION IF NOT ON
 4998                                          .EQUIV  BIT6,WBIT       ;LABEL 'WBIT' IS EQUIVALENT TO BIT 6
 4999  034026  012737  000006  172310  20$:    MOV     #00006,KIPDR4   ;PAGE 4 HAS A PAGE LENGTH OF 1
 5000  034034  012737  000600  172350          MOV     #600,KIPAR4     ;MAP PAGE 4 TO 12K
 5001  034042  012737  034332  000250          MOV     #10$,MMVEC      ;SET M.M. TRAP VECTOR TO 10$
 5002  034050  012737  034056  001110          MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 11$
 5003  034056  012737  012345  100000  1$:     MOV     #12345,@#100000 ;WRITE INTO PAGE 4
 5004  034064  013737  172310  001176          MOV     KIPDR4,$TMP0    ;READ PDR 4 TO CHECK W-BIT
 5005  034072  032737  000100  001176          BIT     #WBIT,$TMP0     ;SEE IF W-BIT IS SET
 5006  034100  001001                          BNE     2$              ;BRANCH IF W-BIT IS SET
 5007  034102  104055                          ERROR   55              ;W-BIT NOT SET
 5008  034104  012737  034112  001110  2$:     MOV     #12$,$LPERR     ;SET LOOP ON ERROR POINTER TO 12$
 5009  034112  012737  012345  100100  12$:    MOV     #12345,@#100100 ;WRITE INTO PAGE 4, BLOCK 2
 5010  034120  013737  172310  001176          MOV     KIPDR4,$TMP0    ;READ PDR 4TO SEE IF W-BIT REMAINED SET
 5011  034126  032737  000100  001176          BIT     #WBIT,$TMP0     ;SEE IF W-BIT REMAINED SET
 5012  034134  001001                          BNE     3$              ;BRANCH IF W-BIT IS SET
 5013  034136  104056                          ERROR   56              ;W-BIT DID NOT REMAIN SET
 5014  034140  012737  034146  001110  3$:     MOV     #13$,$LPERR     ;SET LOOP ON ERROR POINTER TO 13$
 5015  034146  112737  000004  172310  13$:    MOVB    #004,KIPDR4     ;CHANGE PAGE 4'S ACCESS CONTROL FIELD
 5016  034154  013737  172310  001176          MOV     KIPDR4,$TMP0    ;READ PDR 4 TO CHECK W-BIT CLEARING
 5017  034162  022737  000004  001176          CMP     #00004,$TMP0    ;SEE IF W-BIT GOT CLEARED ON PDR WRITE
 5018  034170  001401                          BEQ     7$              ;BRANCH IF W-BIT GOT CLEARED
 5019  034172  104057                          ERROR   57              ;W-BIT DIDN'T GET CLEARED ON PDR WRITE
 5020  034174  012737  034202  001110  7$:     MOV     #17$,$LPERR     ;SET LOOP ON ERROR POINTER TO 17$
 5021  034202  013737  177776  001176  17$:    MOV     PSW,$TMP0       ;READ PROCESSOR STATUS WORD
 5022  034210  013737  001176  177776          MOV     $TMP0,PSW       ;WRITE TO PSW TO CHECK 'GOOD DATO L'
 5023  034216  013737  172316  001176          MOV     KIPDR7,$TMP0    ;READ PDR 7 TO CHECK W-BIT
 5024  034224  105737  001450                  TSTB    FORTY           ;EXECUTING ON AN 11/40?
 5025  034230  001406                          BEQ     21$             ;BRANCH IF NO
 5026  034232  032737  000100  001176          BIT     #WBIT,$TMP0     ;DID W-BIT SET FOR 11/40 PSW WRITE?
 5027  034240  001007                          BNE     8$              ;BRANCH IF W-BIT SET
 5028  034242  104060                          ERROR   60              ;W-BIT DIDN'T SET ON 11/40 PSW WRITE
 5029  034244  000405                          BR      8$
 5030  034246                          21$:
 5031  034246  032737  000100  001176          BIT     #WBIT,$TMP0     ;DID W-BIT STAY 0 FOR 11/6X PSW WRITE?
 5032  034254  001401                          BEQ     8$              ;BRANCH IF W-BIT WAS NOT SET
 5033  034256  104060                          ERROR   60              ;W-BIT DID GET SET ON 11/6X
 5034  034260  012737  034266  001110  8$:     MOV     #18$,$LPERR     ;SET LOOP ON ERROR POINTER TO 18$
 5035  034266  012737  077406  172310  18$:    MOV     #77406,KIPDR4   ;LOAD AN INTERNAL REGISTER, SHOULDN'T
 5036                                                                  ;AFFECT PAGE 7'S W-BIT
 5037  034274  013737  172316  001176          MOV     KIPDR7,$TMP0    ;READ PDR 7 TO CHECK W-BIT AGAIN
 5038  034302  032737  000100  001176          BIT     #WBIT,$TMP0     ;SEE IF W-BIT CLEAR
 5039  034310  001401                          BEQ     9$              ;BRANCH IF W-BIT IS STILL CLEAR
 5040  034312  104061                          ERROR   61              ;W-BIT SET AFTER WRITING PDR 4 ON 11/6X
 5041  034314  012737  034026  001110  9$:     MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
 5042  034322  012737  015432  000250          MOV     #MMTRAP,MMVEC   ;RESTORE NORMAL M.M. TRAP ROUTINE
 5043  034330  000404                          BR      TST45           ;;BRANCH TO NEXT TEST
 5044
 5045  034332  042737  160556  177572  10$:    BIC     #160556,MMR0    ;CLEAR MMR0 FOR NEXT READ OR WRITE
 5046  034340  000002                          RTI                     ;RETURN TO TEST AND CHECK A OR W BIT
 5047
 5048
```

```
                                ;*
                                ;*      THESE NEXT TWO (2) TESTS
                                ;*      CHECK FOR DUAL MAPPING
                                ;*      AMONG GROUPS OR INSIDE A GROUP OF PAR/PDR'S.  THIS IS DONE BY
                                ;*      WRITING ONLY ONE PAGE IN A MODE OF OPERATION THEN CHECKING ALL
                                ;*      PDR'S TO SEE THAT ONLY THE ONE UNDER TEST HAS ITS W
                                ;*      BIT SET.  THIS TEST IS RUN IN ALL MODES,
                                ;*      SO THAT ALL THE REGISTER
                                ;*      PAIRS ARE TESTED.
                                ;*
                                ;;*****************************************************************
                                ;*TEST 45       DUAL MAPPING KERNEL MODE SPACE
                                ;*
                                ;*      THIS TEST STARTS BY LOADING ALL THE P.D.R.'S WITH 77406
                                ;*      (4K PAGE,READ/WRITE).  THEN THE VIRTUAL ADDRESS IS SET TO
                                ;*      010200 AND THAT WORD IS WRITTEN INTO ITSELF.  THIS WRITE WILL
                                ;*      SET THE W BIT IN THE KERNEL SPACE P.D.R. UNDER TEST.  NOW ALL OF THE
                                ;*      P.D.R.'S ARE COMPARED WITH 77706 AND ANY THAT MATCH, EXCEPT THE
                                ;*      ONE THAT IS UNDER TEST, ARE REPORTED AS DUAL MAPPING ERRORS.
                                ;*      KERNEL PAR7 IS MAPPED TO 12K AND KERNEL PAR6 TO I/O PAGE DURING THE WRITE SINCE
                                ;*      WRITING TO REGISTERS INTERNAL TO THE CPU ON THE 11/6X WILL NOT SET THE W BIT
                                ;*      AND ANY OTHER EXSISTING ADDRESS CAN'T BE GUARANTEED FOR THE I/O PAGE
                                ;*
                                ;;*****************************************************************
034342                          TST45:
034342  000004                          SCOPE
034344  012737  035240  001446          MOV     #TST46,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
                                                                ;TEST FOR ESCAPE ON PARITY ERRORS
                                ;       THE FOLLOWING  CODE IS USED TO INITIALIZE THE PAR'S FOR
                                ;       THE NEXT TESTS, SO THAT THE CODE WILL RUN WITH MEMORY
                                ;       MANAGEMENT FULLY ENABLED.
034352  012737  000000  172340          MOV     #0,KIPAR0       ;
034360  012737  000200  172342          MOV     #200,KIPAR1     ;
034366  012737  000400  172344          MOV     #400,KIPAR2     ;
034374  012737  000600  172346          MOV     #600,KIPAR3     ;
034402  012737  000600  172350          MOV     #600,KIPAR4     ;
034410  012737  000600  172352          MOV     #600,KIPAR5     ;
034416  012737  000600  172354          MOV     #600,KIPAR6     ;
034424  012737  007600  172356          MOV     #7600,KIPAR7    ;
034432  012737  000000  177640          MOV     #0,UIPAR0       ;
034440  012737  000200  177642          MOV     #200,UIPAR1     ;
034446  012737  000400  177644          MOV     #400,UIPAR2     ;
034454  012737  000600  177646          MOV     #600,UIPAR3     ;
034462  012737  000600  177650          MOV     #600,UIPAR4     ;
034470  012737  000600  177652          MOV     #600,UIPAR5     ;
034476  012737  000600  177654          MOV     #600,UIPAR6     ;
034504  012737  007600  177656          MOV     #7600,UIPAR7    ;
034512  012737  034522  001110          MOV     #20$,$LPERR     ;SET LOOP ON ERROR POINTER TO 20$
034520  104410                          TBITO                   ;MAKE SURE THAT T-BIT IS OFF FOR
                                                                ;THE DUAL MAPPING TESTS
034522  012737  034612  001110  20$:    MOV     #21$,$LPERR     ;SET LOOP ON ERROR POINTER TO 21$
034530  012737  000000  177776          MOV     #00000,PSW      ;GO TO KERNEL MODE
034536  012703  172300                  MOV     #KIPDR0,R3      ;LOAD FIRST ADDRESS OF KERNEL PDR'S
                                                                ;THAT WILL BE TESTED IN THIS TEST
034542  012704  000010                  MOV     #10,R4          ;TEST THE NEXT EIGHT PDR'S
```

```
5105  034546  012705  010200              MOV     #10200,R5       ;LOAD STARTING VIRTUAL ADDRESS INTO R5
5106  034552  012700  077406        19$:  MOV     #77406,R0       ;ALL PAGES WILL BE ACF=6
5107  034556  005037  001176              CLR     $TMP0           ;CLEAR CORRECT PDR SET INDICATOR
5108  034562  012702  000010              MOV     #10,R2          ;SET COUNT TO LOAD 8 ADDRESSES
5109  034566  012701  172300              MOV     #KIPDR0,R1      ;PUT ADDRESS OF FIRST PDR IN R1
5110  034572  010021              1$:     MOV     R0,(R1)+        ;LOAD R0 INTO PDR ADDRESSED BY R1
5111  034574  077202                      SOB     R2,1$           ;BRANCH BACK TO 1$ IF R2 IS NOT ZERO
5112  034576  012702  000010              MOV     #10,R2          ;SET COUNT TO LOAD 8 ADDRESSES
5113  034602  012701  177600              MOV     #UIPDR0,R1      ;PUT ADDRESS OF FIRST PDR IN R1
5114  034606  010021              2$:     MOV     R0,(R1)+        ;LOAD R0 INTO PDR ADDRESSED BY R1
5115  034610  077202                      SOB     R2,2$           ;BRANCH BACK TO 2$ IF R2 IS NOT ZERO
5116  034612  012700  077406        21$:  MOV     #77406,R0       ;MUST RE-INIT THESE PDRS IF ERROR
5117  034616  010037  172300              MOV     R0,KIPDR0       ;LOAD KERNEL PDR0
5118  034622  010037  172302              MOV     R0,KIPDR1       ;LOAD KERNEL PDR1
5119  034626  010037  172304              MOV     R0,KIPDR2       ;LOAD KERNEL PDR2
5120  034632  010037  172316              MOV     R0,KIPDR7       ;LOAD KERNEL PDR7
5121  034636  010037  172300              MOV     R0,KIPDR0               ;LOAD PDR0 OF PRESENT SPACE
5122  034642  010037  172316              MOV     R0,KIPDR7               ;LOAD PDR7 OF PRESENT SPACE
5123  034646  020527  170200              CMP     R5,#170200      ;IS PDR7 BEING TESTED?
5124  034652  001032                      BNE     22$             ;BRANCH IF IT ISN'T
5125  034654  012737  007600  172354      MOV     #7600,KIPAR6    ;MAP PAR6 TO THE I/O PAGE
5126  034662  012737  000600  172356      MOV     #600,KIPAR7     ;RE-MAP PAR7 TO 12K
5127  034670  011515                      MOV     (R5),(R5)       ;NOW WRITE USING PAR7
5128  034672  042737  000001  157572      BIC     #BIT0,@#157572  ;TURN OFF MEMORY MANAGMENT USING
5129                                                              ;PAR6-ADDRESS OF MMR0
5130  034700  022737  077506  172316      CMP     #77506,KIPDR7   ;CHECK PDR 7 BEFORE RESTORING PDR/PAR'S
5131  034706  001005                      BNE     24$             ;DID W-BIT GET SET?
5132                                                              ;BRANCH IF NO
5133  034710  005237  001176              INC     $TMP0           ;SET CORRECT PDR SET INDICATOR
5134  034714  012737  077406  172300      MOV     #77406,KIPDR0   ;RESTORE PDR0 AFTER WRITING $TMP0
5135  034722  012737  077406  172314  24$:MOV     #77406,KIPDR6   ;RESET PDR6
5136  034730  012737  000600  172354      MOV     #600,KIPAR6     ;RE-MAP PAR6 TO 12K
5137  034736  000401                      BR      23$             ;AND NOW COMPARE TO OTHER PDR'S
5138  034740  011515              22$:    MOV     (R5),(R5)       ;WRITE INTO PAGE UNDER TEST
5139  034742                      23$:
5140  034742  012702  000010              MOV     #10,R2          ;SET COUNTER TO READ NEXT 10 REGISTERS
5141  034746  012701  172300              MOV     #KIPDR0,R1      ;LOAD ADDRESS OF BEGINNING PDR
5142  034752  011100              7$:     MOV     (R1),R0         ;READ PDR INTO R0
5143  034754  022700  077506              CMP     #77506,R0       ;SEE IF THIS WAS THE PDR WITH ITS W BIT ON
5144  034760  001030                      BNE     9$              ;BRANCH IF THIS IS NOT THE ONE
5145  034762  020103                      CMP     R1,R3           ;SEE IF THE ADDRESS MATCHES PDR UNDER TEST
5146  034764  001424                      BEQ     8$              ;BRANCH IF ADDRESS IS CORRECT
5147  034766  104062                      ERROR   62              ;W BIT GOT SET IN WRONG PDR
5148  034770  012700  077406              MOV     #77406,R0       ;RE-SET PAGES MODIFIED BY ERROR
5149  034774  010037  172300              MOV     R0,KIPDR0       ;RELOAD KERNEL PDR0
5150  035000  010037  172302              MOV     R0,KIPDR1       ;RELOAD KERNEL PDR1
5151  035004  010037  172304              MOV     R0,KIPDR2       ;RELOAD KERNEL PDR2
5152  035010  010037  172316              MOV     R0,KIPDR7       ;RELOAD KERNEL PDR7
5153  035014  010037  172300              MOV     R0,KIPDR0               ;RELOAD PAGE 0 OF PRESENT SPACE
5154  035020  010037  172316              MOV     R0,KIPDR7               ;RE-LOAD I/O PAGE PDR IF ERROR
5155  035024  020527  170200              CMP     R5,#170200      ;SEE IF TESTING PDR7
5156  035030  001404                      BEQ     9$              ;DON'T WRITE IF TESTING PDR7
5157  035032  011515                      MOV     (R5),(R5)       ;TRY WRITE AGAIN, IN CASE YOU
5158                                                              ;WERE NOT TESTING PAGE SEVEN
5159  035034  000402                      BR      9$              ;GO UPDATE R1 FOR NEXT READ
5160  035036  005237  001176        8$:   INC     $TMP0           ;SET FLAG SINCE ADDRESSES MATCHED
```

```
5161  035042  062701  000002          9$:     ADD     #2,R1           ;POINT TO NEXT PDR TO BE READ
5162  035046  077237                  SOB     R2,7$           ;BRANCH TO 7$ IF ALL PDR'S NOT READ
5163  035050  012702  000010          MOV     #10,R2          ;SET COUNTER TO READ NEXT 10 REGISTERS
5164  035054  012701  177600          MOV     #UIPDR0,R1      ;LOAD ADDRESS OF BEGINNING PDR
5165  035060  011100                  13$:    MOV     (R1),R0         ;READ PDR INTO R0
5166  035062  022700  077506          CMP     #77506,R0       ;SEE IF THIS WAS THE PDR WITH ITS W BIT ON
5167  035066  001030                  BNE     15$             ;BRANCH IF THIS IS NOT THE ONE
5168  035070  020103                  CMP     R1,R3           ;SEE IF THE ADDRESS MATCHES PDR UNDER TEST
5169  035072  001424                  BEQ     14$             ;BRANCH IF ADDRESS IS CORRECT
5170  035074  104062                  ERROR   62              ;W BIT GOT SET IN WRONG PDR
5171  035076  012700  077406          MOV     #77406,R0       ;RE-SET PAGES MODIFIED BY ERROR
5172  035102  010037  172300          MOV     R0,KIPDR0       ;RELOAD KERNEL PDR0
5173  035106  010037  172302          MOV     R0,KIPDR1       ;RELOAD KERNEL PDR1
5174  035112  010037  172304          MOV     R0,KIPDR2       ;RELOAD KERNEL PDR2
5175  035116  010037  172316          MOV     R0,KIPDR7       ;RELOAD KERNEL PDR7
5176  035122  010037  172300          MOV     R0,KIPDR0               ;RELOAD PAGE 0 OF PRESENT SPACE
5177  035126  010037  172316          MOV     R0,KIPDR7               ;RE-LOAD I/O PAGE PDR IF ERROR
5178  035132  020527  170200          CMP     R5,#170200      ;SEE IF TESTING PDR7
5179  035136  001404                  BEQ     15$             ;DON'T WRITE IF TESTING PDR7
5180  035140  011515                  MOV     (R5),(R5)       ;TRY WRITE AGAIN, IN CASE YOU
5181                                                          ;WERE NOT TESTING PAGE SEVEN
5182  035142  000402                  BR      15$             ;GO UPDATE R1 FOR NEXT READ
5183  035144  005237  001176          14$:    INC     $TMP0           ;SET FLAG SINCE ADDRESSES MATCHED
5184  035150  062701  000002          15$:    ADD     #2,R1           ;POINT TO NEXT PDR TO BE READ
5185  035154  077237                  SOB     R2,13$          ;BRANCH TO 13$ IF ALL PDR'S NOT READ
5186  035156  005737  001176          TST     $TMP0           ;SEE IF THERE WAS A CORRECT PDR
5187  035162  001002                  BNE     16$             ;BRANCH IF THERE WAS
5188  035164  011300                  MOV     (R3),R0         ;SAVE CONTENTS OF PDR UNDER TEST
5189  035166  104063                  ERROR   63              ;NO PDR ADDRESSES MATCHED
5190  035170  062703  000002          16$:    ADD     #2,R3           ;POINT TO NEXT PDR UNDER TEST
5191  035174  062705  020000          ADD     #20000,R5       ;CHANGE PAGE NUMBER IN VIRT. ADDR.
5192  035200  005304                  DEC     R4              ;DECREMENT COUNTER
5193  035202  001402                  BEQ     17$             ;BRANCH IF COUNTER IS ZERO
5194  035204  000137  034552          JMP     19$             ;JUMP TO LOAD PDR'S AGAIN
5195  035210  012737  007600  172356  17$:    MOV     #7600,KIPAR7    ;REMAP PAR7 TO THE I/O PAGE
5196  035216  052737  000001  177572  BIS     #BIT0,MMR0      ;REENABLE MEMORY MANAGEMENT AFTER
5197                                                          ;TESTING PDR7
5198  035224  012737  000340  177776  MOV     #340,PSW        ;RETURN TO KERNEL MODE, PRIORITY 7
5199  035232  012737  034522  001110  MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
5200
5201
5202
5203
5204                                  ;;*****************************************************************
5205                                  ;*TEST 46       DUAL MAPPING USER MODE SPACE
5206                                  ;;
5207                                  ;;      THIS TEST STARTS BY LOADING ALL THE P.D.R.'S WITH 77406
5208                                  ;;      (4K PAGE,READ/WRITE).  THEN THE VIRTUAL ADDRESS IS SET TO
5209                                  ;;      010200 AND THAT WORD IS WRITTEN INTO ITSELF.  THIS WRITE WILL
5210                                  ;;      SET THE W BIT IN THE USER SPACE P.D.R. UNDER TEST.  NOW ALL OF THE
5211                                  ;;      P.D.R.'S ARE COMPARED WITH 77706 AND ANY THAT MATCH, EXCEPT THE
5212                                  ;;      ONE THAT IS UNDER TEST, ARE REPORTED AS DUAL MAPPING ERRORS.
5213                                  ;;      USER PAR7 IS MAPPED TO 12K AND USER PAR6 TO I/O PAGE DURING THE WRITE SINCE
5214                                  ;;      WRITING TO REGISTERS INTERNAL TO THE CPU ON THE 11/6X WILL NOT SET THE W BIT
5215                                  ;;      AND ANY OTHER EXSISTING ADDRESS CAN'T BE GUARANTEED FOR THE I/O PAGE
5216                                  ;;
```

```
                                ;;*****************************************************************
035240                          ;TST46:
035240  000004                          SCOPE
035242  012737  035766  001446          MOV     #TST47,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
                                                                ;TEST FOR ESCAPE ON PARITY ERRORS
035250  012737  035340  001110  20$:     MOV     #21$,$LPERR     ;SET LOOP ON ERROR POINTER TO 21$
035256  012737  140000  177776          MOV     #140000,PSW     ;GO TO USER MODE
035264  012703  177600                  MOV     #UIPDR0,R3      ;LOAD FIRST ADDRESS OF USER PDR'S
                                                                ;THAT WILL BE TESTED IN THIS TEST
035270  012704  000010                  MOV     #10,R4          ;TEST THE NEXT EIGHT PDR'S
035274  012705  010200                  MOV     #10200,R5       ;LOAD STARTING VIRTUAL ADDRESS INTO R5
035300  012700  077406          19$:    MOV     #77406,R0       ;ALL PAGES WILL BE ACF=6
035304  005037  001176                  CLR     $TMP0           ;CLEAR CORRECT PDR SET INDICATOR
035310  012702  000010                  MOV     #10,R2          ;SET COUNT TO LOAD 8 ADDRESSES
035314  012701  172300                  MOV     #KIPDR0,R1      ;PUT ADDRESS OF FIRST PDR IN R1
035320  010021          1$:     MOV     R0,(R1)+        ;LOAD R0 INTO PDR ADDRESSED BY R1
035322  077202                  SOB     R2,1$           ;BRANCH BACK TO 1$ IF R2 IS NOT ZERO
035324  012702  000010                  MOV     #10,R2          ;SET COUNT TO LOAD 8 ADDRESSES
035330  012701  177600                  MOV     #UIPDR0,R1      ;PUT ADDRESS OF FIRST PDR IN R1
035334  010021          2$:     MOV     R0,(R1)+        ;LOAD R0 INTO PDR ADDRESSED BY R1
035336  077202                  SOB     R2,2$           ;BRANCH BACK TO 2$ IF R2 IS NOT ZERO
035340  012700  077406          21$:    MOV     #77406,R0       ;MUST RE-INIT THESE PDRS IF ERROR
035344  010037  172300                  MOV     R0,KIPDR0       ;LOAD KERNEL PDR0
035350  010037  172302                  MOV     R0,KIPDR1       ;LOAD KERNEL PDR1
035354  010037  172304                  MOV     R0,KIPDR2       ;LOAD KERNEL PDR2
035360  010037  172316                  MOV     R0,KIPDR7       ;LOAD KERNEL PDR7
035364  010037  177600                  MOV     R0,UIPDR0               ;LOAD PDR0 OF PRESENT SPACE
035370  010037  177616                  MOV     R0,UIPDR7               ;LOAD PDR7 OF PRESENT SPACE
035374  020527  170200                  CMP     R5,#170200      ;IS PDR7 BEING TESTED?
035400  001032                  BNE     22$             ;BRANCH IF IT ISN'T
035402  012737  007600  177654          MOV     #7600,UIPAR6    ;MAP PAR6 TO THE I/O PAGE
035410  012737  000600  177656          MOV     #600,UIPAR7     ;RE-MAP PAR7 TO 12K
035416  011515                  MOV     (R5),(R5)       ;NOW WRITE USING PAR7
035420  042737  000001  157572          BIC     #BIT0,@#157572  ;TURN OFF MEMORY MANAGMENT USING
                                                                ;PAR6-ADDRESS OF MMR0
035426  022737  077506  177616          CMP     #77506,UIPDR7   ;CHECK PDR 7 BEFORE RESTORING PDR/PAR'S
035434  001005                  BNE     24$             ;DID W-BIT GET SET?
                                                                ;BRANCH IF NO
035436  005237  001176                  INC     $TMP0           ;SET CORRECT PDR SET INDICATOR
035442  012737  077406  177600          MOV     #77406,UIPDR0   ;RESTORE PDR0 AFTER WRITING $TMP0
035450  012737  077406  177614  24$:    MOV     #77406,UIPDR6   ;RESET PDR6
035456  012737  000600  177654          MOV     #600,UIPAR6     ;RE-MAP PAR6 TO 12K
035464  000401                  BR      23$             ;AND NOW COMPARE TO OTHER PDR'S
035466  011515          22$:    MOV     (R5),(R5)       ;WRITE INTO PAGE UNDER TEST
035470                  23$:
035470  012702  000010                  MOV     #10,R2          ;SET COUNTER TO READ NEXT 10 REGISTERS
035474  012701  172300                  MOV     #KIPDR0,R1      ;LOAD ADDRESS OF BEGINNING PDR
035500  011100          7$:     MOV     (R1),R0         ;READ PDR INTO R0
035502  022700  077506                  CMP     #77506,R0       ;SEE IF THIS WAS THE PDR WITH ITS W BIT ON
035506  001030                  BNE     9$              ;BRANCH IF THIS IS NOT THE ONE
035510  020103                  CMP     R1,R3           ;SEE IF THE ADDRESS MATCHES PDR UNDER TEST
035512  001424                  BEQ     8$              ;BRANCH IF ADDRESS IS CORRECT
035514  104062                  ERROR   62              ;W BIT GOT SET IN WRONG PDR
035516  012700  077406                  MOV     #77406,R0       ;RE-SET PAGES MODIFIED BY ERROR
035522  010037  172300                  MOV     R0,KIPDR0       ;RELOAD KERNEL PDR0
035526  010037  172302                  MOV     R0,KIPDR1       ;RELOAD KERNEL PDR1
```

```
5273   035532  010037  172304                  MOV     R0,KIPDR2       ;RELOAD KERNEL PDR2
5274   035536  010037  172316                  MOV     R0,KIPDR7       ;RELOAD KERNEL PDR7
5275   035542  010037  177600                  MOV     R0,UIPDR0               ;RELOAD PAGE 0 OF PRESENT SPACE
5276   035546  010037  177616                  MOV     R0,UIPDR7               ;RE-LOAD I/O PAGE PDR IF ERROR
5277   035552  020527  170200                  CMP     R5,#170200      ;SEE IF TESTING PDR7
5278   035556  001404                          BEQ     9$              ;DON'T WRITE IF TESTING PDR7
5279   035560  011515                          MOV     (R5),(R5)       ;TRY WRITE AGAIN, IN CASE YOU
5280                                                                   ;WERE NOT TESTING PAGE SEVEN
5281   035562  000402                          BR      9$              ;GO UPDATE R1 FOR NEXT READ
5282   035564  005237  001176          8$:     INC     $TMP0           ;SET FLAG SINCE ADDRESSES MATCHED
5283   035570  062701  000002          9$:     ADD     #2,R1           ;POINT TO NEXT PDR TO BE READ
5284   035574  077237                          SOB     R2,7$           ;BRANCH TO 7$ IF ALL PDR'S NOT READ
5285   035576  012702  000010                  MOV     #10,R2          ;SET COUNTER TO READ NEXT 10 REGISTERS
5286   035602  012701  177600                  MOV     #UIPDR0,R1      ;LOAD ADDRESS OF BEGINNING PDR
5287   035606  011100                  13$:    MOV     (R1),R0         ;READ PDR INTO R0
5288   035610  022700  077506                  CMP     #77506,R0       ;SEE IF THIS WAS THE PDR WITH ITS W BIT ON
5289   035614  001030                          BNE     15$             ;BRANCH IF THIS IS NOT THE ONE
5290   035616  020103                          CMP     R1,R3           ;SEE IF THE ADDRESS MATCHES PDR UNDER TEST
5291   035620  001424                          BEQ     14$             ;BRANCH IF ADDRESS IS CORRECT
5292   035622  104062                          ERROR   62              ;W BIT GOT SET IN WRONG PDR
5293   035624  012700  077406                  MOV     #77406,R0       ;RE-SET PAGES MODIFIED BY ERROR
5294   035630  010037  172300                  MOV     R0,KIPDR0       ;RELOAD KERNEL PDR0
5295   035634  010037  172302                  MOV     R0,KIPDR1       ;RELOAD KERNEL PDR1
5296   035640  010037  172304                  MOV     R0,KIPDR2       ;RELOAD KERNEL PDR2
5297   035644  010037  172316                  MOV     R0,KIPDR7       ;RELOAD KERNEL PDR7
5298   035650  010037  177600                  MOV     R0,UIPDR0               ;RELOAD PAGE 0 OF PRESENT SPACE
5299   035654  010037  177616                  MOV     R0,UIPDR7               ;RE-LOAD I/O PAGE PDR IF ERROR
5300   035660  020527  170200                  CMP     R5,#170200      ;SEE IF TESTING PDR7
5301   035664  001404                          BEQ     15$             ;DON'T WRITE IF TESTING PDR7
5302   035666  011515                          MOV     (R5),(R5)       ;TRY WRITE AGAIN, IN CASE YOU
5303                                                                   ;WERE NOT TESTING PAGE SEVEN
5304   035670  000402                          BR      15$             ;GO UPDATE R1 FOR NEXT READ
5305   035672  005237  001176          14$:    INC     $TMP0           ;SET FLAG SINCE ADDRESSES MATCHED
5306   035676  062701  000002          15$:    ADD     #2,R1           ;POINT TO NEXT PDR TO BE READ
5307   035702  077237                          SOB     R2,13$          ;BRANCH TO 13$ IF ALL PDR'S NOT READ
5308   035704  005737  001176                  TST     $TMP0           ;SEE IF THERE WAS A CORRECT PDR
5309   035710  001002                          BNE     16$             ;BRANCH IF THERE WAS
5310   035712  011300                          MOV     (R3),R0         ;SAVE CONTENTS OF PDR UNDER TEST
5311   035714  104063                          ERROR   63              ;NO PDR ADDRESSES MATCHED
5312   035716  062703  000002          16$:    ADD     #2,R3           ;POINT TO NEXT PDR UNDER TEST
5313   035722  062705  020000                  ADD     #20000,R5       ;CHANGE PAGE NUMBER IN VIRT. ADDR.
5314   035726  005304                          DEC     R4              ;DECREMENT COUNTER
5315   035730  001402                          BEQ     17$             ;BRANCH IF COUNTER IS ZERO
5316   035732  000137  035300                  JMP     19$             ;JUMP TO LOAD PDR'S AGAIN
5317   035736  012737  007600  177656  17$:    MOV     #7600,UIPAR7    ;REMAP PAR7 TO THE I/O PAGE
5318   035744  052737  000001  177572          BIS     #BIT0,MMR0      ;REENABLE MEMORY MANAGEMENT AFTER
5319                                                                   ;TESTING PDR7
5320   035752  012737  000340  177776          MOV     #340,PSW        ;RETURN TO KERNEL MODE, PRIORITY 7
5321   035760  012737  035250  001110          MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
5322
5323
5324
5325
5326                                   .SBTTL  ************ ENTRY POINT 7 --- STARTING ADDRESS 230 ************
5327                                   .SBTTL  ***** MOVE FROM AND MOVE TO PREVIOUS MODE INSTRUCTION TEST *****
5328                                   ;*
```

```
                                ;*      THIS GROUP OF TESTS WILL TEST ALL THE LOGIC ASSOCIATED WITH
                                ;*      THE "MOVE FROM PREVIOUS" AND MOVE TO PREVIOUS" INSTRUCTIONS.
                                ;*
                                ;*
                                ;*
                                ;*
                                ;*
                                ;*TEST 47       MOVE FROM PREVIOUS (USER) I-SPACE
                                ;*
                                ;*      THIS TEST USES THE 'MFPI' INSTRUCTION TO ENSURE THAT THE
                                ;*      PREVIOUS MODE IS CLOCKED CORRECTLY
                                ;*      THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE TESTED,
                                ;*
                                ;*
                                ;*      IF THE CORRECT MODE IS NOT ENABLED A NON-RESIDENT ABORT
                                ;*      WILL OCCUR AND TRAP TO 10$, WHERE THE ERRORS ARE REPORTED.
                                ;*
035766                          TST47:
035766  000004                          SCOPE
035770  012737  036730  001446          MOV     #TST50,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
                                                                ;TEST FOR ESCAPE ON PARITY ERRORS
035776  104411                          TBITR                   ;RESTORE T-BIT IF IT WAS ON BEFORE
                                                                ;THE DUAL MAPPING TESTS
036000  012737  036232  001106  ENTPT7: MOV     #20$,$LPADR     ;SET LOOP ADDRESS POINTER TO 20$
036006  012737  036232  001110          MOV     #20$,$LPERR     ;SET LOOP ON ERROR POINTER TO 20$
036014  012737  000047  001102          MOV     #47,$TSTNM      ;LOAD TEST NUMBER INTO MEMORY
036022  013777  001102  143112          MOV     $TSTNM,@DISPLAY ;DISPLAY TEST NUMBER FOR THIS TEST
036030  012737  077406  172300          MOV     #77406,KIPDR0   ;MAKE KERNEL I PAGE 0 200 BLOCKS, R/W
036036  012737  077406  172302          MOV     #77406,KIPDR1   ;MAKE KERNEL I PAGE 1 200 BLOCKS, R/W
036044  012737  077406  172304          MOV     #77406,KIPDR2   ;MAKE KERNEL I PAGE 2 200 BLOCKS, R/W
036052  012737  077406  172306          MOV     #77406,KIPDR3   ;MAKE KERNEL I PAGE 3 200 BLOCKS, R/W
036060  012737  077406  172316          MOV     #77406,KIPDR7   ;MAKE KERNEL I PAGE 7 200 BLOCKS, R/W
036066  012737  000000  172340          MOV     #000,KIPAR0     ;MAP KERNEL I PAGE 0 TO 0 - 4K
036074  012737  000200  172342          MOV     #200,KIPAR1     ;MAP KERNEL I PAGE 1 TO 4K - 8K
036102  012737  000400  172344          MOV     #400,KIPAR2     ;MAP KERNEL I PAGE 2 TO 8K - 12K
036110  012737  000600  172346          MOV     #600,KIPAR3     ;MAP KERNEL I PAGE 3 TO 12K - 16K
036116  012737  007600  172356          MOV     #7600,KIPAR7    ;MAP KERNEL I PAGE 7 TO THE I/O PAGE
036124  012737  000001  177572          MOV     #BIT0,MMR0      ;ENABLE 18-BIT RELOCATION IF NOT ON
036132  012700  077406                  MOV     #77406,R0       ;MAKE ALL KERNEL I-SPACE PAGES RESIDENT
                                                                ;READ/WRITE, LENGTH 200 BLOCKS
036136  012702  000010                  MOV     #10,R2          ;SET COUNT TO LOAD 8 ADDRESSES
036142  012701  172300                  MOV     #KIPDR0,R1      ;PUT ADDRESS OF FIRST PDR IN R1
036146  010021                  19$:    MOV     R0,(R1)+        ;LOAD R0 INTO PDR ADDRESSED BY R1
036150  077202                          SOB     R2,19$          ;BRANCH BACK TO 19$ IF R2 IS NOT ZERO
036152  012702  000010                  MOV     #10,R2          ;SET COUNT TO LOAD 8 ADDRESSES
036156  012701  177600                  MOV     #UIPDR0,R1      ;PUT ADDRESS OF FIRST PDR IN R1
036162  010021                  30$:    MOV     R0,(R1)+        ;LOAD R0 INTO PDR ADDRESSED BY R1
036164  077202                          SOB     R2,30$          ;BRANCH BACK TO 30$ IF R2 IS NOT ZERO
036166  012737  000000  177640          MOV     #000,UIPAR0     ;MAP USER I PAGE 0 TO 0-4K
036174  012737  000200  177642          MOV     #200,UIPAR1     ;MAP USER I PAGE 1 TO 4-8K
036202  012737  000400  177644          MOV     #400,UIPAR2     ;MAP USER I PAGE 2 TO 8-12K
036210  012737  000600  177646          MOV     #600,UIPAR3     ;MAP USER I PAGE 3 TO 12-16K
036216  012737  007600  177656          MOV     #7600,UIPAR7    ;MAP USER I PAGE 7 TO THE I/O PAGE
036224  012737  036232  001110          MOV     #20$,$LPERR     ;SET LOOP ON ERROR TO 20$
036232                          20$:
036232  012737  077406  172310          MOV     #77406,KIPDR4   ;KERNEL I-SPACE PAGE 4 READ/WRITE
```

```
5385  036240  012737  000600  172350          MOV     #600,KIPAR4     ;MAP KERNEL I PAGE 4 TO 12K
5386  036246  012737  000600  177650          MOV     #600,UIPAR4     ;MAP USER I PAGE 4 TO 12K
5387  036254  012700  036514                  MOV     #36514,R0       ;LOAD DATA PATTERN INTO R0
5388  036260  010037  100000                  MOV     R0,@#100000     ;LOAD DATA PATTERN INTO PHY 60000
5389  036264  012737  036662  000250          MOV     #10$,MMVEC      ;SET M.M. VECTOR TO 10$
5390  036272  105037  172310                  CLRB    KIPDR4          ;MAKE KERNEL I-SPACE PAGE 4 NON-RESIDENT
5391                                          ;THE FOLLOWING WILL TEST DSTM=0 MFPI
5392                                          ;
5393  036276  012737  036304  001110          MOV     #11$,$LPERR     ;SET LOOP ON ERROR POINTER TO 11$
5394  036304  012737  030340  177776  11$:    MOV     #030340,PSW     ;MAKE PREVIOUS MODE USER
5395  036312  006506                  1$:     MFPI    USP             ;PUT USER STACK POINTER ON KERNEL
5396                                                                  ;STACK
5397  036314  022706  001100                  CMP     #KERSTK,KSP     ;WAS SOMETHING PUSHED ON STACK AT 1$
5398  036320  001407                          BEQ     3$              ;BRANCH IF NOTHING WAS PUSHED
5399  036322  012601                          MOV     (KSP)+,R1       ;POP KERNEL STACK INTO R1
5400  036324  012702  000700                  MOV     #USESTK,R2      ;EXPECTING TO GET 700 AS USP
5401  036330  020201                          CMP     R2,R1           ;DID YOU GET THE RIGHT POINTER?
5402  036332  001403                          BEQ     2$              ;BRANCH IF YOU DID
5403  036334  104064                          ERROR   64              ;WRONG THING WAS PUSHED ON STACK
5404  036336  000401                          BR      2$              ;BRANCH TO NEXT TRY
5405  036340  104065                  3$:     ERROR   65              ;NOTHING PUSHED ON STACK
5406  036342                          2$:     ;THE FOLLOWING WILL TEST DSTM=1 MFPI.
5407  036342  012737  036350  001110          MOV     #12$,$LPERR     ;SET LOOP ON ERROR POINTER TO 12$
5408  036350  012737  030340  177776  12$:    MOV     #030340,PSW     ;MAKE PREVIOUS MODE USER
5409  036356  012702  100000                  MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
5410  036362  006512                          MFPI    (R2)            ;READ FROM PHYSICAL 60000
5411  036364  012601                          MOV     (KSP)+,R1       ;POP KERNEL STACK INTO R1
5412  036366  020001                          CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
5413  036370  001401                          BEQ     4$              ;BRANCH IF CORRECT DATA WAS FETCHED
5414  036372  104066                          ERROR   66              ;WRONG DATA WAS FETCHED
5415  036374                          4$:     ;THE FOLLOWING WILL TEST DSTM=2 MFPI.
5416  036374  012737  036402  001110          MOV     #14$,$LPERR     ;SET LOOP ON ERROR POINTER TO 14$
5417  036402  012737  030340  177776  14$:    MOV     #030340,PSW     ;MAKE PREVIOUS MODE USER
5418  036410  012702  100000                  MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
5419  036414  006522                          MFPI    (R2)+           ;READ FROM PHYSICAL 60000
5420  036416  012601                          MOV     (KSP)+,R1       ;POP KERNEL STACK INTO R1
5421  036420  020001                          CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
5422  036422  001401                          BEQ     5$              ;BRANCH IF CORRECT DATA WAS FETCHED
5423  036424  104066                          ERROR   66              ;WRONG DATA WAS FETCHED
5424  036426                          5$:     ;THE FOLLOWING WILL TEST DSTM=3 MFPI.
5425  036426  012737  036434  001110          MOV     #15$,$LPERR     ;SET LOOP ON ERROR POINTER TO 15$
5426  036434  012737  030340  177776  15$:    MOV     #030340,PSW     ;MAKE PREVIOUS MODE USER
5427  036442  006537  100000                  MFPI    @#100000        ;READ FROM PHYSICAL 60000
5428  036446  012601                          MOV     (KSP)+,R1       ;POP KERNEL STACK INTO R1
5429  036450  020001                          CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
5430  036452  001401                          BEQ     6$              ;BRANCH IF CORRECT DATA WAS FETCHED
5431  036454  104066                          ERROR   66              ;WRONG DATA WAS FETCHED
5432  036456                          6$:     ;THE FOLLOWING WILL TEST DSTM=4 MFPI.
5433  036456  012737  036464  001110          MOV     #16$,$LPERR     ;SET LOOP ON ERROR POINTER TO 16$
5434  036464  012737  030340  177776  16$:    MOV     #030340,PSW     ;MAKE PREVIOUS MODE USER
5435  036472  012702  100002                  MOV     #100002,R2      ;LOAD VIRTUAL ADDRESS INTO R2
5436  036476  006542                          MFPI    -(R2)           ;READ FROM PHYSICAL 60000
5437  036500  012601                          MOV     (KSP)+,R1       ;POP KERNEL STACK INTO R1
5438  036502  020001                          CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
5439  036504  001401                          BEQ     7$              ;BRANCH IF CORRECT DATA WAS FETCHED
5440  036506  104066                          ERROR   66              ;WRONG DATA WAS FETCHED
```

```
5441  036510                          7$:
5442                                  ;THE FOLLOWING WILL TEST DSTM=5 MFPI.
5443                                  ;
5444  036510  012737  036516  001110  MOV     #17$,$LPERR     ;SET LOOP ON ERROR POINTER TO 17$
5445  036516  012737  030340  177776  17$:    MOV     #030340,PSW     ;MAKE PREVIOUS MODE USER
5446  036524  012737  100000  001202  MOV     #100000,$TMP2   ;LOAD TEST LOC. VIRT. ADDR INTO LOC. $TMP2
5447  036532  012702  001204          MOV     #<$TMP2+2>,R2   ;LOAD ADDR. OF $TMP2+2 INTO R2
5448  036536  006552                  MFPI    @-(R2)          ;READ FROM PHYSICAL 60000
5449  036540  012601                  MOV     (KSP)+,R1       ;POP KERNEL STACK INTO R1
5450  036542  020001                  CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
5451  036544  001401                  BEQ     8$              ;BRANCH IF CORRECT DATA WAS FETCHED
5452  036546  104066                  ERROR   66              ;WRONG DATA WAS FETCHED
5453  036550                          8$:     ;THE FOLLOWING WILL TEST DSTM=6 MFPI.
5454                                  ;
5455  036550  012737  036556  001110  MOV     #18$,$LPERR     ;SET LOOP ON ERROR POINTER TO 18$
5456  036556  012737  030340  177776  18$:    MOV     #030340,PSW     ;MAKE PREVIOUS MODE USER
5457  036564  005002                  CLR     R2              ;MAKE REGISTER 2 A ZERO
5458  036566  006562  100000          MFPI    100000(R2)      ;READ FROM PHYSICAL 60000
5459  036572  012601                  MOV     (KSP)+,R1       ;POP KERNEL STACK INTO R1
5460  036574  020001                  CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
5461  036576  001401                  BEQ     9$              ;BRANCH IF CORRECT DATA WAS FETCHED
5462  036600  104066                  ERROR   66              ;WRONG DATA WAS FETCHED
5463  036602                          9$:     ;THE FOLLOWING WILL TEST DSTM=7 MFPI.
5464                                  ;
5465  036602  012737  036610  001110  MOV     #22$,$LPERR     ;SET LOOP ON ERROR POINTER TO 22$
5466  036610  012737  030340  177776  22$:    MOV     #030340,PSW     ;MAKE PREVIOUS MODE USER
5467  036616  012737  100000  001202  MOV     #100000,$TMP2   ;LOAD TEST LOC. V.A. INTO $TMP2
5468  036624  012702  001202          MOV     #$TMP2,R2       ;LOAD ADDRESS OF $TMP2 INTO R2
5469  036630  006572  000000          MFPI    @0(R2)          ;USE $TMP2 TO FETCH VIRTUAL
5470                                                          ;ADDRESS OF 60000
5471  036634  012601                  MOV     (KSP)+,R1       ;POP KERNEL STACK INTO R1
5472  036636  020001                  CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
5473  036640  001401                  BEQ     25$             ;BRANCH IF CORRECT DATA WAS FETCHED
5474  036642  104066                  ERROR   66              ;WRONG DATA WAS FETCHED
5475  036644  012737  015432  000250  25$:    MOV     #MMTRAP,MMVEC   ;SET M.M. VECTOR TO NORMAL ROUTINE
5476  036652  012737  036232  001110  MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
5477  036660  000423                  BR      TST50           ;;BRANCH TO NEXT TEST
5478
5479
5480  036662  012637  001436          10$:    MOV     (KSP)+,OLDPC    ;SAVE PC & PS OF TRAP
5481  036666  012637  001440                  MOV     (KSP)+,OLDPS
5482  036672  013737  177572  001404          MOV     MMR0,PMMR0      ;SAVE MMR0 FOR ERROR TYPEOUT
5483  036700  013737  177576  001406          MOV     MMR2,PMMR2      ;SAVE MMR2 FOR ERROR TYPEOUT
5484  036706  042737  160000  177572          BIC     #160000,MMR0    ;CLEAR ERROR BITS IN MMR0
5485  036714  104067                          ERROR   67              ;TRIED TO READ NON-RESIDENT PAGE
5486  036716  013746  001440                  MOV     OLDPS,-(KSP)    ;PUT PC & PS OF TRAP ON STACK
5487  036722  013746  001436                  MOV     OLDPC,-(KSP)
5488  036726  000002                          RTI
5489
5490
5491                                  ;;**********************************************************************
5492                                  ;*TEST 50       MOVE TO PREVIOUS (USER) I-SPACE
5493                                  ;*
5494                                  ;*      THIS TEST USES THE 'MTPI' INSTRUCTION TO ENSURE THAT THE
5495                                  ;*      PREVIOUS MODE IS CLOCKED CORRECTLY
5496                                  ;*      THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE TESTED,
```

```
5497                                            ;*
5498                                            ;*
5499                                            ;*      IF THE CORRECT MODE IS NOT ENABLED A NON-RESIDENT ABORT
5500                                            ;*      WILL OCCUR AND TRAP TO 10$, WHERE THE ERRORS ARE REPORTED.
5501                                            ;*
5502                                            ;;**********************************************************************
5503  036730                                    TST50:
5504  036730  000004                                   SCOPE
5505  036732  012737  037616  001446                    MOV     #TST51,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
5506                                                                            ;TEST FOR ESCAPE ON PARITY ERRORS
5507  036740  012737  077406  172310  20$:      MOV     #77406,KIPDR4   ;KERNEL I-SPACE PAGE 4 READ/WRITE
5508  036746  012737  077406  177610            MOV     #77406,UIPDR4   ;USER I-SPACE PAGE 4 READ/WRITE
5509  036754  012737  000600  172350            MOV     #600,KIPAR4     ;MAP KERNEL I PAGE 4 TO 12K
5510  036762  012737  000600  177650            MOV     #600,UIPAR4     ;MAP USER I PAGE 4 TO 12K
5511  036770  012737  037550  000250            MOV     #10$,MMVEC      ;SET M.M. VECTOR TO 10$
5512                                            ;THE FOLLOWING WILL TEST DSTM=0 MTPI
5513                                            ;
5514  036776  012737  030340  177776  1$:       MOV     #030340,PSW     ;MAKE PREVIOUS MODE USER
5515  037004  012746  007777                    MOV     #7777,-(KSP)    ;PUSH DATA ON KERNEL STACK
5516  037010  006606                            MTPI    USP             ;LOAD USER STACK POINTER
5517  037012  006506                            MFPI    USP             ;READ USER STACK POINTER
5518  037014  012601                            MOV     (KSP)+,R1       ;POP KERNEL STACK INTO R1
5519  037016  022701  007777                    CMP     #7777,R1        ;WAS USER STACK POINTER CHANGED
5520  037022  001401                            BEQ     2$              ;BRANCH IF IT WAS
5521  037024  104070                            ERROR   70              ;USER STACK POINTER NOT CHANGED
5522  037026  012737  030340  177776  2$:       MOV     #030340,PSW     ;MAKE PREVIOUS MODE USER
5523  037034  012746  000700                    MOV     #USESTK,-(KSP)  ;GET READY TO RESTORE USER S. POINT
5524  037040  006606                            MTPI    USP             ;RESTORE USER STACK POINTER
5525  037042                            3$:     ;THIS WILL TEST DSTM = 1 MTPI.
5526  037042  012737  037060  001110            MOV     #13$,$LPERR     ;SET LOOP ON ERROR POINTER TO 13$
5527  037050  012702  100000                    MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
5528  037054  012700  125252                    MOV     #125252,R0      ;LOAD TEST DATA INTO R0
5529  037060  010046                    13$:    MOV     R0,-(KSP)       ;PUSH TEST DATA ON KERNEL STACK
5530  037062  105037  172310                    CLRB    KIPDR4          ;MAKE KERNEL I PAGE 4 NON-RESIDENT
5531  037066  006612                            MTPI    (R2)            ;LOAD TEST DATA INTO PHYSICAL 60000
5532  037070  112737  000006  172310            MOVB    #006,KIPDR4     ;MAKE KERNEL PAGE 4 RESIDENT
5533  037076  011201                            MOV     (R2),R1         ;READ FROM ADDRESS 60000
5534  037100  020001                            CMP     R0,R1           ;SEE IF DATA WAS STORED AT CORRECT PLACE
5535  037102  001401                            BEQ     4$              ;BRANCH IF STORE WAS CORRECT
5536  037104  104071                            ERROR   71              ;INCORRECT STORE
5537  037106                            4$:     ;THIS WILL TEST DSTM = 3 MTPI.
5538  037106  012737  037126  001110            MOV     #14$,$LPERR     ;SET LOOP ON ERROR POINTER TO 14$
5539  037114  012737  030340  177776            MOV     #030340,PSW     ;MAKE PREVIOUS MODE USER
5540  037122  012700  052525                    MOV     #52525,R0       ;LOAD TEST DATA INTO R0
5541  037126  010046                    14$:    MOV     R0,-(KSP)       ;PUSH TEST DATA ON KERNEL STACK
5542  037130  105037  172310                    CLRB    KIPDR4          ;MAKE KERNEL I PAGE 4 NON-RESIDENT
5543  037134  006637  100000                    MTPI    @#100000        ;LOAD TEST DATA INTO PHYSICAL 60000
5544  037140  112737  000006  172310            MOVB    #006,KIPDR4     ;MAKE KERNEL PAGE 4 RESIDENT
5545  037146  013701  100000                    MOV     @#100000,R1     ;READ FROM ADDRESS 60000
5546  037152  020001                            CMP     R0,R1           ;SEE IF DATA WAS STORED CORRECTLY
5547  037154  001401                            BEQ     5$              ;BRANCH IF STORE WAS CORRECT
5548  037156  104071                            ERROR   71              ;INCORRECT STORE
5549  037160                            5$:     ;THIS WILL TEST DSTM = 4 MTPI.
5550  037160  012737  037200  001110            MOV     #15$,$LPERR     ;SET LOOP ON ERROR POINTER TO 15$
5551  037166  012737  030340  177776            MOV     #030340,PSW     ;MAKE PREVIOUS MODE USER
5552  037174  012700  125252                    MOV     #125252,R0      ;LOAD TEST DATA INTO R0
```

```
5553  037200  010046                    15$:  MOV     R0,-(KSP)       ;PUSH TEST DATA ON KERNEL STACK
5554  037202  012702  100002                  MOV     #100002,R2      ;LOAD VIRTUAL ADDRESS INTO R2
5555  037206  105037  172310                  CLRB    KIPDR4          ;MAKE KERNEL I PAGE 4 NON-RESIDENT
5556  037212  006642                          MTPI    -(R2)           ;LOAD TEST DATA INTO PHYSICAL 60000
5557  037214  112737  000006  172310          MOVB    #006,KIPDR4     ;MAKE KERNEL PAGE 4 RESIDENT
5558  037222  013701  100000                  MOV     @#100000,R1     ;READ FROM ADDRESS 60000
5559  037226  020001                          CMP     R0,R1           ;SEE IF DATA WAS STORED CORRECTLY
5560  037230  001401                          BEQ     6$              ;BRANCH IF STORE WAS CORRECT
5561  037232  104071                          ERROR   71              ;INCORRECT STORE
5562  037234                              6$:   ;THIS WILL TEST DSTM = 6 MTPI. BELOW ARE THE
5563                                            ;
5564  037234  012737  037256  001110          MOV     #16$,$LPERR     ;SET LOOP ON ERROR POINTER TO 16$
5565  037242  012737  030340  177776          MOV     #030340,PSW     ;MAKE PREVIOUS MODE USER
5566  037250  012700  052525                  MOV     #52525,R0       ;LOAD TEST DATA INTO R0
5567  037254  005002                          CLR     R2              ;MAKE REGISTER 2 ZERO
5568  037256  010046                    16$:  MOV     R0,-(KSP)       ;PUSH TEST DATA ON KERNEL STACK
5569  037260  105037  172310                  CLRB    KIPDR4          ;MAKE KERNEL I PAGE 4 NON-RESIDENT
5570  037264  006662  100000                  MTPI    100000(R2)      ;LOAD TEST DATA INTO PHYSICAL 60000
5571  037270  112737  000006  172310          MOVB    #006,KIPDR4     ;MAKE KERNEL PAGE 4 RESIDENT
5572  037276  013701  100000                  MOV     @#100000,R1     ;READ FROM ADDRESS 60000
5573  037302  020001                          CMP     R0,R1           ;SEE IF DATA WAS STORED CORRECTLY
5574  037304  001401                          BEQ     7$              ;BRANCH IF STORE WAS CORRECT
5575  037306  104071                          ERROR   71              ;INCORRECT STORE
5576  037310                              7$:   ;THE FOLLOWING WILL TEST DSTM=2 MTPI.
5577                                            ;
5578  037310  012737  037334  001110          MOV     #17$,$LPERR     ;SET LOOP ON ERROR POINTER TO 17$
5579  037316  012737  030340  177776          MOV     #030340,PSW     ;MAKE PREVIOUS MODE USER
5580  037324  012700  125252                  MOV     #125252,R0      ;LOAD TEST DATA INTO R0
5581  037330  012702  100000                  MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
5582  037334  010046                    17$:  MOV     R0,-(KSP)       ;PUSH TEST DATA ON KERNEL STACK
5583  037336  105037  172310                  CLRB    KIPDR4          ;MAKE KERNEL PAGE 4 NON-RESIDENT
5584  037342  006612                          MTPI    (R2)            ;LOAD TEST DATA INTO PHYSICAL 60000
5585  037344  112737  000006  172310          MOVB    #006,KIPDR4     ;MAKE KERNEL PAGE 4 RESISENT
5586  037352  013701  100000                  MOV     @#100000,R1     ;READ FROM ADDRESS 60000
5587  037356  020001                          CMP     R0,R1           ;SEE IF DATA WAS STORED CORRECTLY
5588  037360  001401                          BEQ     8$              ;BRANCH IF STORE WAS CORRECT
5589  037362  104071                          ERROR   71              ;INCORRECT STORE
5590  037364                              8$:   ;THE FOLLOWING WILL TEST DSTM=5 MTPI.
5591                                            ;
5592  037364  012737  037416  001110          MOV     #18$,$LPERR     ;SET LOOP ON ERROR POINTER TO 18$
5593  037372  012737  030340  177776          MOV     #030340,PSW     ;MAKE PREVIOUS MODE USER
5594  037400  012700  052525                  MOV     #52525,R0       ;LOAD TEST DATA INTO R0
5595  037404  012702  001204                  MOV     #<$TMP2+2>,R2   ;LOAD ADDR. OF LOC. $TMP2+2 INTO R2
5596  037410  012737  100000  001202          MOV     #100000,$TMP2   ;LOAD VIRT. ADDR. OF TEST LOC. INTO $TMP2
5597  037416  010046                    18$:  MOV     R0,-(KSP)       ;PUSH TEST DATA ON KERNEL STACK
5598  037420  105037  172310                  CLRB    KIPDR4          ;MAKE KERNEL PAGE 4 NON-RESIDENT
5599  037424  006652                          MTPI    @-(R2)          ;LOAD TEST DATA INTO PHYSICAL 60000
5600  037426  112737  000006  172310          MOVB    #006,KIPDR4     ;MAKE KERNEL PAGE 4 RESIDENT
5601  037434  013701  100000                  MOV     @#100000,R1     ;READ FROM ADDRESS 60000
5602  037440  020001                          CMP     R0,R1           ;SEE IF DATA WAS STORED CORRECTLY
5603  037442  001401                          BEQ     9$              ;BRANCH IF STORE WAS CORRECT
5604  037444  104071                          ERROR   71              ;INCORRECT STORE
5605  037446                              9$:   ;THE FOLLOWING WILL TEST DSTM=7 MTPI.
5606                                            ;
5607  037446  012737  037500  001110          MOV     #19$,$LPERR     ;SET LOOP ON ERROR POINTER TO 19$
5608  037454  012737  030340  177776          MOV     #030340,PSW     ;MAKE PREVIOUS MODE USER
```

```
5609  037462  012700  125252              MOV     #125252,R0      ;LOAD TEST DATA INTO R0
5610  037466  012737  100000  001202      MOV     #100000,$TMP2   ;LOAD VIRT. ADDR. OF TEST LOCATION
5611                                                              ;INTO LOCATION $TMP2
5612  037474  012702  001202              MOV     #$TMP2,R2       ;LOAD ADDRESS OF $TMP2 INTO R2
5613  037500  010046              19$:    MOV     R0,-(KSP)       ;PUSH TEST DATA ON KERNEL STACK
5614  037502  105037  172310              CLRB    KIPDR4          ;MAKE KERNEL PAGE 4 NON-RESIDENT
5615  037506  006672  000000              MTPI    @0(R2)          ;LOAD TEST DATA INTO PHYSICAL 60000
5616  037512  112737  000006  172310      MOVB    #006,KIPDR4     ;MAKE KERNEL PAGE 4 RESIDENT
5617  037520  013701  100000              MOV     @#100000,R1     ;READ FROM ADDRESS 60000
5618  037524  020001                      CMP     R0,R1           ;SEE IF DATA WAS STORED CORRECTLY
5619  037526  001401                      BEQ     25$             ;BRANCH IF STORE WAS CORRECT
5620  037530  104071                      ERROR   71              ;INCORRECT STORE
5621  037532  012737  036740  001110  25$:  MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
5622  037540  012737  015432  000250      MOV     #MMTRAP,MMVEC   ;RESTORE M.M. VECTOR TO NORMAL ROUTINE
5623  037546  000423                      BR      TST51           ;;BRANCH TO NEXT TEST
5624
5625
5626  037550  012637  001436  10$:    MOV     (KSP)+,OLDPC    ;SAVE PC & PS OF TRAP
5627  037554  012637  001440            MOV     (KSP)+,OLDPS
5628  037560  013737  177572  001404    MOV     MMR0,PMMR0      ;SAVE MMR0 FOR ERROR TYPEOUT
5629  037566  013737  177576  001406    MOV     MMR2,PMMR2      ;SAVE MMR2 FOR ERROR TYPEOUT
5630  037574  042737  160000  177572    BIC     #160000,MMR0    ;CLEAR ERROR BITS IN MMR0
5631  037602  104067                    ERROR   67              ;TRIED TO LOAD A N.R. PAGE 4
5632  037604  013746  001440            MOV     OLDPS,-(KSP)    ;PUT PC & PS OF TRAP ON STACK
5633  037610  013746  001436            MOV     OLDPC,-(KSP)
5634  037614  000002                    RTI                     ;RETURN TO TEST
5635
5636
5637
5638                                ;;**************************************************************
5639                                ;*TEST 51       MOVE FROM PREVIOUS (KERNEL) I-SPACE TO USER MODE
5640                                ;*
5641                                ;*      THIS TEST CHECKS THAT IF THE PREVIOUS MODE IS KERNEL THE
5642                                ;*       FETCH IS FROM
5643                                ;*      KERNEL MODE.
5644                                ;*      THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE TESTED,
5645                                ;*
5646                                ;*
5647                                ;*      IF THE CORRECT MODE IS NOT ENABLED A NON-RESIDENT ABORT
5648                                ;*      WILL OCCUR AND TRAP TO 10$, WHERE THE ERRORS ARE REPORTED.
5649                                ;*
5650                                ;;**************************************************************
5651  037616                        TST51:
5652  037616  000004                        SCOPE
5653  037620  012737  040364  001446        MOV     #TST52,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
5654                                                                ;TEST FOR ESCAPE ON PARITY ERRORS
5655  037626  012700  077406                MOV     #77406,R0       ;MAKE ALL USER I-SPACE PAGES RESIDENT
5656                                                                ;READ/WRITE, LENGTH 200 BLOCKS
5657  037632  012702  000010                MOV     #10,R2          ;SET COUNT TO LOAD 8 ADDRESSES
5658  037636  012701  177600                MOV     #UIPDR0,R1      ;PUT ADDRESS OF FIRST PDR IN R1
5659  037642  010021                19$:    MOV     R0,(R1)+        ;LOAD R0 INTO PDR ADDRESSED BY R1
5660  037644  077202                        SOB     R2,19$          ;BRANCH BACK TO 19$ IF R2 IS NOT ZERO
5661  037646  012737  037654  001110        MOV     #20$,$LPERR     ;SET LOOP ON ERROR TO 20$
5662  037654  012737  140340  177776  20$:  MOV     #140340,PSW     ;GO TO USER MODE FOR THIS TEST
5663  037662  012737  077406  172310        MOV     #77406,KIPDR4   ;KERNEL I-SPACE PAGE 4 READ/WRITE
5664  037670  012737  000600  172350        MOV     #600,KIPAR4     ;MAP KERNEL I PAGE 4 TO 12K
```

```
5665  037676  012737  000600  177650          MOV     #600,UIPAR4     ;MAP USER I PAGE 4 TO 12K
5666  037704  012700  036514                  MOV     #36514,R0       ;LOAD DATA PATTERN INTO R0
5667  037710  010037  100000                  MOV     R0,@#100000     ;LOAD DATA PATTERN INTO PHY 60000
5668  037714  012702  100000                  MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
5669                                          ;THE FOLLOWING WILL TEST DSTM=0 MFPI
5670                                          ;
5671  037720  012737  040316  000250          MOV     #10$,MMVEC      ;SET M.M. VECTOR TO 10$
5672  037726  105037  177610                  CLRB    UIPDR4          ;MAKE USER I-SPACE PAGE 4 NON-RESIDENT
5673  037732  012737  140340  177776          MOV     #140340,PSW     ;MAKE PREVIOUS MODE KERNEL PRESENT USER
5674  037740  006506                  1$:     MFPI    KSP             ;PUT KERNEL STACK POINTER ON USER STACK
5675  037742  022706  000700                  CMP     #USESTK,USP     ;WAS SOMETHING PUSHED ON STACK AT 1$
5676  037746  001407                          BEQ     3$              ;BRANCH IF NOTHING WAS PUSHED
5677  037750  012601                          MOV     (USP)+,R1       ;POP USER STACK INTO R1
5678  037752  012702  001100                  MOV     #KERSTK,R2      ;EXPECTING 1100 AS KSP
5679  037756  020201                          CMP     R2,R1           ;DID YOU GET THE RIGHT POINTER?
5680  037760  001403                          BEQ     2$              ;BRANCH IF YOU DID
5681  037762  104064                          ERROR   64              ;WRONG THING WAS PUSHED ON STACK
5682  037764  000401                          BR      2$              ;BRANCH TO NEXT TRY
5683  037766  104065                  3$:     ERROR   65              ;NOTHING PUSHED ON STACK
5684  037770                          2$:     ;THE FOLLOWING WILL TEST DSTM=1 MFPI.
5685  037770  012737  037776  001110          MOV     #12$,$LPERR     ;SET LOOP ON ERROR POINTER TO 12$
5686  037776  012737  140340  177776  12$:    MOV     #140340,PSW     ;MAKE PREVIOUS MODE KERNEL PRESENT USER
5687  040004  012702  100000                  MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
5688  040010  006512                          MFPI    (R2)            ;READ FROM PHYSICAL 60000
5689  040012  012601                          MOV     (USP)+,R1       ;POP USER STACK INTO R1
5690  040014  020001                          CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
5691  040016  001401                          BEQ     4$              ;BRANCH IF CORRECT DATA WAS FETCHED
5692  040020  104066                          ERROR   66              ;WRONG DATA WAS FETCHED
5693  040022                          4$:     ;THE FOLLOWING WILL TEST DSM=2 MFPI.
5694  040022  012737  040030  001110          MOV     #14$,$LPERR     ;SET LOOP ON ERROR POINTER TO 14$
5695  040030  012737  140340  177776  14$:    MOV     #140340,PSW     ;MAKE PREVIOUS MODE KERNEL PRESENT USER
5696  040036  012702  100000                  MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
5697  040042  006522                          MFPI    (R2)+           ;READ FROM PHYSICAL 60000
5698  040044  012601                          MOV     (USP)+,R1       ;POP USER STACK INTO R1
5699  040046  020001                          CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
5700  040050  001401                          BEQ     5$              ;BRANCH IF CORRECT DATA WAS FETCHED
5701  040052  104066                          ERROR   66              ;WRONG DATA WAS FETCHED
5702  040054                          5$:     ;THE FOLLOWING WILL TEST DSTM=3 MFPI.
5703  040054  012737  040062  001110          MOV     #15$,$LPERR     ;SET LOOP ON ERROR POINTER TO 15$
5704  040062  012737  140340  177776  15$:    MOV     #140340,PSW     ;MAKE PREVIOUS MODE KERNEL PRESENT USER
5705  040070  006537  100000                  MFPI    @#100000        ;READ FROM PHYSICAL 60000
5706  040074  012601                          MOV     (USP)+,R1       ;POP USER STACK INTO R1
5707  040076  020001                          CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
5708  040100  001401                          BEQ     6$              ;BRANCH IF CORRECT DATA WAS FETCHED
5709  040102  104066                          ERROR   66              ;WRONG DATA WAS FETCHED
5710  040104                          6$:     ;THE FOLLOWING WILL TEST DSTM=4 MFPI.
5711  040104  012737  040112  001110          MOV     #16$,$LPERR     ;SET LOOP ON ERROR POINTER TO 16$
5712  040112  012737  140340  177776  16$:    MOV     #140340,PSW     ;MAKE PREVIOUS MODE DERNEL PRESENT USER
5713  040120  012702  100002                  MOV     #100002,R2      ;LOAD VIRTUAL ADDRESS INTO R2
5714  040124  006542                          MFPI    -(R2)           ;READ FROM PHYSICAL 60000
5715  040126  012601                          MOV     (USP)+,R1       ;POP USER STACK INTO R1
5716  040130  020001                          CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
5717  040132  001401                          BEQ     7$              ;BRANCH IF CORRECT DATA WAS FETCHED
5718  040134  104066                          ERROR   66              ;WRONG DATA WAS FETCHED
5719  040136                          7$:     ;THE FOLLOWING WILL TEST DSTM=5 MFPI.
5720                                          ;
```

```
5721  040136  012737  040144  001110          MOV     #17$,$LPERR     ;SET LOOP ON ERROR POINTER TO 17$
5722  040144  012737  140340  177776  17$:    MOV     #140340,PSW     ;MAKE PREVIOUS MODE KERNEL PRESENT USER
5723  040152  012737  100000  001202          MOV     #100000,$TMP2   ;LOAD TEST LOC. VIRT. ADDR INTO LOC. $TMP2
5724  040160  012702  001204                  MOV     #<$TMP2+2>,R2   ;LOAD ADDRESS OF $TMP2+2 INTO R2
5725  040164  006552                          MFPI    @-(R2)          ;READ FROM PHYSICAL 60000
5726  040166  012601                          MOV     (USP)+,R1       ;POP USER STACK INTO R1
5727  040170  020001                          CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
5728  040172  001401                          BEQ     8$              ;BRANCH IF CORRECT DATA FETCHED
5729  040174  104066                          ERROR   66              ;WRONG DATA WAS FETCHED
5730  040176                          8$:     ;THE FOLLOWING WILL TEST DSTM=6 MFPI.
5731                                          ;
5732  040176  012737  040204  001110          MOV     #18$,$LPERR     ;SET LOOP ON ERROR POINTER TO 18$.
5733  040204  012737  140340  177776  18$:    MOV     #140340,PSW     ;MAKE PREVIOUS MODE KERNEL PRESENT USER
5734  040212  005002                          CLR     R2              ;MAKE REGISTER 2 A ZERO
5735  040214  006562  100000                  MFPI    100000(R2)      ;READ FROM PHYSICAL 60000
5736  040220  012601                          MOV     (USP)+,R1       ;POP USER STACK INTO R1
5737  040222  020001                          CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
5738  040224  001401                          BEQ     9$              ;BRANCH IF CORRECT DATA FETCHED
5739  040226  104066                          ERROR   66              ;WRONG DATA WAS FETCHED
5740  040230                          9$:     ;THE FOLLOWING WILL TEST DSTM=7 MFPI.
5741                                          ;
5742  040230  012737  040236  001110          MOV     #22$,$LPERR     ;SET LOOP ON ERROR POINTER TO 22$
5743  040236  012737  140340  177776  22$:    MOV     #140340,PSW     ;MAKE PREVIOUS MODE KERNEL PRESENT USER
5744  040244  012737  100000  001202          MOV     #100000,$TMP2   ;LOAD TEST LOC. VIRT. ADDR. INTO $TMP2
5745  040252  012702  001202                  MOV     #$TMP2,R2       ;LOAD ADDRESS OF $TMP2 INTO R2
5746  040256  006572  000000                  MFPI    @0(R2)          ;READ FROM PHYSICAL 60000
5747  040262  012601                          MOV     (USP)+,R1       ;POP USER STACK INTO R1
5748  040264  020001                          CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
5749  040266  001401                          BEQ     25$             ;BRANCH IF CORRECT DATA FETCHED
5750  040270  104066                          ERROR   66              ;WRONG DATA WAS FETCHED
5751  040272  012737  015432  000250  25$:    MOV     #MMTRAP,MMVEC   ;SET M.M. VECTOR TO NORMAL ROUTINE
5752  040300  012737  000340  177776          MOV     #00340,PSW      ;GO BACK TO KERNEL MODE, PREVIOUS KERNEL
5753  040306  012737  037654  001110          MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
5754  040314  000423                          BR      TST52           ;;BRANCH TO NEXT TEXT
5755
5756
5757  040316  012637  001436          10$:    MOV     (KSP)+,OLDPC    ;SAVE PC & PS OF TRAP
5758  040322  012637  001440                  MOV     (KSP)+,OLDPS
5759  040326  013737  177572  001404          MOV     MMR0,PMMR0      ;SAVE MMR0 FOR ERROR TYPEOUT
5760  040334  013737  177576  001406          MOV     MMR2,PMMR2      ;SAVE MMR2 FOR ERROR TYPEOUT
5761  040342  042737  160000  177572          BIC     #160000,MMR0    ;CLEAR ERROR BITS IN MMR0
5762  040350  104067                          ERROR   67              ;TRIED TO READ NON-RESIDENT PAGE
5763  040352  013746  001440                  MOV     OLDPS,-(KSP)    ;PUT PC & PS OF TRAP ON STACK
5764  040356  013746  001436                  MOV     OLDPC,-(KSP)
5765  040362  000002                          RTI                     ;RETURN TO TEST
5766
5767                                  ;;**************************************************************
5768                                  ;*TEST 52       MOVE FROM/TO D-SPACE = MOVE FROM/TO I-SPACE
5769                                  ;*
5770                                  ;*      THIS TEST CHECKS THAT SINCE THERE IS NO DISTINCTION
5771                                  ;*      BETWEEN INSTRUCTION AND DATA SPACE IN THE 11/6X & 11/40
5772                                  ;*      MFPD & MTPD SHOULD BE DECODED THE SAME AS MFPI & MTPI.
5773                                  ;*
5774                                  ;;**************************************************************
5775  040364                          TST52:
5776  040364  000004                          SCOPE
```

```
5777  040366  012737  040476  001446          MOV     #TST53,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
5778                                                                    ;TEST FOR ESCAPE ON PARITY ERRORS
5779  040374  012737  030340  177776  20$:    MOV     #030340,PSW     ;MAKE PREVIOUS MODE=USER,CURRENT=KERNEL
5780  040402  106506                          MFPD    USP             ;MFPD SHOULD ACT LIKE MFPI PUTTING
5781                                                                    ;USER STACK POINTER ON THE KERNEL STACK
5782  040404  022706  001100                  CMP     #KERSTK,KSP     ;WAS SOMETHING PUSHED ON KERNEL STACK?
5783  040410  001407                          BEQ     2$              ;BRANCH IF NO
5784  040412  012601                          MOV     (KSP)+,R1       ;POP KERNEL STACK INTO R1
5785  040414  012702  000700                  MOV     #USESTK,R2      ;EXPECTING TO GET 700 AS USP
5786  040420  020201                          CMP     R2,R1           ;DID GET RIGHT POINTER VALUE?
5787  040422  001403                          BEQ     3$              ;BRANCH IF YES
5788  040424  104064                          ERROR   64              ;WRONG THING WAS PUSHED ON STACK
5789  040426  000401                          BR      3$              ;BRANCH TO NEXT TRY
5790  040430  104065                  2$:     ERROR   65              ;NOTHING PUSHED ON STACK
5791  040432  012737  040440  001110  3$:     MOV     #4$,$LPERR      ;SET LOOP ON ERROR POINTER TO 4$
5792  040440  012746  007777          4$:     MOV     #7777,-(KSP)    ;PUSH DATA ON KERNEL STACK
5793  040444  106606                          MTPD    USP             ;LOAD THE USER STACK POINTER
5794  040446  106506                          MFPD    USP             ;READ USER STACK POINTER
5795  040450  012601                          MOV     (KSP)+,R1       ;POP KERNEL STACK INTO R1
5796  040452  022701  007777                  CMP     #7777,R1        ;WAS USER STACK POINTER CHANGED?
5797  040456  001401                          BEQ     5$              ;BRANCH IF YES
5798  040460  104070                          ERROR   70              ;USER STACK POINTER NOT CHANGED
5799  040462  012746  000700          5$:     MOV     #USESTK,-(KSP)  ;GET READY TO RESTORE USER STK. PTR.
5800  040466  106606                          MTPD    USP             ;RESTORE USER STACK POINTER
5801  040470  012737  040374  001110          MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
5802
5803                                  ;;**************************************************************
5804                                  ;*TEST 53       MOVE FROM PREVIOUS I=SPACE (PREVIOUS=CURRENT=KERNEL)
5805                                  ;*
5806                                  ;*      THIS TEST CHECKS THAT IF BOTH PREVIOUS AND CURRENT MODES
5807                                  ;*      ARE KERNEL, AND THE SOURCE MODE IS 0, THE DESTINATION
5808                                  ;*      STACK IS NOT DECREMENTED BEFORE ACCESS.
5809                                  ;*      (FOR 11/40 THIS IS PURPOSE OF 'HOOK MUX')
5810                                  ;*      "MFPI KSP" SHOULD PUSH THE NON-DECREMENTED VALUE
5811                                  ;*      OF KSP (1100) ONTO THE STACK (AT LOC. 1076).
5812                                  ;;**************************************************************
5813  040476  000004                  TST53:  SCOPE
5814  040500  012737  040554  001446          MOV     #$EOP,NXTTST    ;SET ESCAPE POINTER TO EOP ROUTINE
5815  040506  012737  040514  001110          MOV     #20$,$LPERR     ;SET LOOP ON ERROR POINTER TO 20$
5816  040514  005037  177776          20$:    CLR     @#PSW           ;SET PREVIOUS = CURRENT = KERNEL
5817  040520  012700  001100                  MOV     #STACK,R0       ;SETUP VALUE FOR STACK POINTER
5818  040524  010006                          MOV     R0,KSP          ;LOAD STACK POINTER
5819  040526  006506                          MFPI    KSP             ;THE VALUE "STACK" SHOULD BE PUSHED
5820                                                                  ;BEFORE BEING DECREMENTED
5821  040530  011601                          MOV     (KSP),R1        ;READ DATA WHICH WAS PUSHED
5822  040532  020001                          CMP     R0,R1           ;WAS THE ORIGINAL VALUE OF THE
5823                                                                  ;STACK POINTER PUSHED?
5824  040534  001401                          BEQ     1$              ;BRANCH IF YES
5825  040536  104066                          ERROR   66              ;MFPI FETCHED WRONG DATA
5826  040540  005740                  1$:     TST     -(R0)           ;SETUP EXPECTED STACK POINTER VALUE
5827  040542  020600                          CMP     KSP,R0          ;WAS THE STACK POINTER DECREMENTED?
5828  040544  001401                          BEQ     2$              ;BRANCH IF YES
5829  040546  104065                          ERROR   65              ;STACK NOT PUSHED BY THE MFPI
5830  040550  012706  001100          2$:     MOV     #STACK,KSP      ;RESTORE STACK POINTER
5831
5832
```

```
                                .SBTTL  ************************************************************************

                                .SBTTL  END OF PASS ROUTINE

                                ;;**********************************************************************
                                ;*INCREMENT THE PASS NUMBER ($PASS)
                                ;*TYPE "END PASS #XXXXX TOTAL NUMBER OF ERRORS SINCE LAST REPORT YYYYY"
                                ;*WHERE XXXXX AND YYYYY ARE DECIMAL NUMBERS
                                ;*IF SW12=1 INHIBIT TRACE TRAP
                                ;*IF THERES A MONITOR GO TO IT
                                ;*IF THERE ISN'T JUMP TO LOOP

040554                          $EOP:
040554  000004                          SCOPE
040556  005037  001102                  CLR     $TSTNM          ;;ZERO THE TEST NUMBER
040562  005037  001212                  CLR     $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
040566  005237  001234                  INC     $PASS           ;;INCREMENT THE PASS NUMBER
040572  042737  100000  001234          BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
040600  005327                          DEC     (PC)+           ;;LOOP?
040602  000001                  $EOPCT: .WORD   1
040604  003072                          BGT     $DOAGN          ;;YES
040606  012737                          MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
040610  000001                  $ENDCT: .WORD   1
040612  040602                          $EOPCT
040614  104401  040622                  TYPE    ,65$            ;;TYPE ASCIZ STRING
040620  000407                          BR      64$             ;;GET OVER THE ASCIZ
                                ;;65$:  .ASCIZ  <12><15>/END PASS #/
040640                          64$:
040640  013746  001234                  MOV     $PASS,-(SP)     ;;SAVE $PASS FOR TYPEOUT
                                                                ;;TYPE PASS NUMBER
040644  104405                          TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
040646  104401  040654                  TYPE    ,67$            ;;TYPE ASCIZ STRING
040652  000421                          BR      66$             ;;GET OVER THE ASCIZ
                                ;;67$:  .ASCIZ  /  TOTAL ERRORS SINCE LAST REPORT /
040716                          66$:
040716  013746  001112                  MOV     $ERTTL,-(SP)    ;;SAVE $ERTTL FOR TYPEOUT
                                                                ;;TOTAL NUMBER OF ERRORS
040722  104405                          TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
040724  104401  001223                  TYPE    ,$CRLF          ;;TYPE CARRIAGE RETURN, LINE FEED
040730  005037  001112                  CLR     $ERTTL          ;;CLEAR ERROR TOTAL
040734  013700  000042          $GET42: MOV     @#42,R0         ;;GET MONITOR ADDRESS
040740  001414                          BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
040742  005046                          CLR     -(SP)           ;;INSURE THE "T" BIT IS CLEAR
040744  012746  040752                  MOV     #$CLR.T,-(SP)   ;;SETUP FOR AN RTI OR RTT
040750  000426                          BR      $RTRN           ;;GO DO AN RTI OR RTT TO LOAD THE PSW
                                                                ;;WITH A CLEARED "T" BIT
040752                          $CLR.T:
040752  013700  000042                  MOV     @#42,R0         ;;INSURE R0 CONTAINS THE MONITORS
040756  001405                          BEQ     $DOAGN          ;;RETURN ADDRESS
040760  000005                          RESET                   ;;CLEAR THE WORLD
040762  004710                  $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
040764  000240                          NOP                     ;;SAVE ROOM
```

```
5889  040766  000240                          NOP                     ;;FOR
5890  040770  000240                          NOP                     ;;ACT11
5891  040772                        $DOAGN:
5892  040772  104400                          TRAP                    ;;PUSH OLD PSW AND PC ON STACK
5893  040774  042716  000020                  BIC     #20,(SP)        ;;CLEAR THE "T" BIT
5894  041000  032777  010000  140132          BIT     #BIT12,@SWR     ;;RUN WITH TRACE TRAP?
5895  041006  001005                          BNE     1$              ;;BR IF NO
5896  041010  005137  041034                  COM     $TBIT           ;;IS IT TIME FOR TRACE TRAP
5897  041014  100402                          BMI     1$              ;;BR IF NO
5898  041016  052716  000020                  BIS     #20,(SP)        ;;SET TRACE TRAP
5899  041022  012746  041030          1$:     MOV     #$LOOP,-(SP)    ;;JUMP TO START OF TEST
5900  041026  000002                $RTRN:  RTI                     ;;RETURN--THIS IS CHANGED TO
5901                                                                ;;AN "RTT" IF "RTT" IS A LEGAL
5902                                                                ;;INSTRUCTION
5903  041030                        $LOOP:
5904  041030  000137                          JMP     @(PC)+          ;;RETURN
5905  041032  020612                $RTNAD: .WORD   LOOP
5906  041034  000000                $TBIT:  .WORD   0               ;;"T" BIT STATE INDICATOR
5907  041036    377     377   000    $ENULL: .BYTE   -1,-1,0         ;;NULL CHARACTER STRING
5908          041042                        .EVEN
5909
5910                                .SBTTL  SAVE AND RESTORE R0-R5 ROUTINES
5911
5912                                ;;**************************************************************
5913                                ;*SAVE R0-R5
5914                                ;*CALL:
5915                                ;*      SAVREG
5916                                ;*UPON RETURN FROM $SAVREG THE STACK WILL LOOK LIKE:
5917                                ;*
5918                                ;*TOP---(+16)
5919                                ;* +2---(+18)
5920                                ;* +4---R5
5921                                ;* +6---R4
5922                                ;* +8---R3
5923                                ;*+10---R2
5924                                ;*+12---R1
5925                                ;*+14---R0
5926
5927  041042                        $SAVREG:
5928  041042  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
5929  041044  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
5930  041046  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
5931  041050  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
5932  041052  010446                          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
5933  041054  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
5934  041056  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PS OF MAIN FLOW
5935  041062  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PC OF MAIN FLOW
5936  041066  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PS OF CALL
5937  041072  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PC OF CALL
5938  041076  000002                          RTI
5939
5940                                ;*RESTORE R0-R5
5941                                ;*CALL:
5942                                ;*      RESREG
5943  041100                        $RESREG:
5944  041100  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PC OF CALL
```

```
5945  041104  012666  000022                MOV     (SP)+,22(SP)    ;;RESTORE PS OF CALL
5946  041110  012666  000022                MOV     (SP)+,22(SP)    ;;RESTORE PC OF MAIN FLOW
5947  041114  012666  000022                MOV     (SP)+,22(SP)    ;;RESTORE PS OF MAIN FLOW
5948  041120  012605                        MOV     (SP)+,R5        ;;POP STACK INTO R5
5949  041122  012604                        MOV     (SP)+,R4        ;;POP STACK INTO R4
5950  041124  012603                        MOV     (SP)+,R3        ;;POP STACK INTO R3
5951  041126  012602                        MOV     (SP)+,R2        ;;POP STACK INTO R2
5952  041130  012601                        MOV     (SP)+,R1        ;;POP STACK INTO R1
5953  041132  012600                        MOV     (SP)+,R0        ;;POP STACK INTO R0
5954  041134  000002                        RTI
5955                          .SBTTL  TYPE ROUTINE
5956
5957                          ;;**********************************************************************
5958                          ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
5959                          ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
5960                          ;*NOTE1:        $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
5961                          ;*NOTE2:        $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
5962                          ;*NOTE3:        $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
5963                          ;*
5964                          ;*CALL:
5965                          ;*1) USING A TRAP INSTRUCTION
5966                          ;*      TYPE    ,MESADR         ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
5967                          ;*OR
5968                          ;*      TYPE
5969                          ;*      MESADR
5970                          ;*
5971
5972  041136  105737  001157  $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
5973  041142  100002                BPL     1$              ;;BR IF YES
5974  041144  000000                HALT                    ;;HALT HERE IF NO TERMINAL
5975  041146  000430                BR      3$              ;;LEAVE
5976  041150  010046          1$:   MOV     R0,-(SP)        ;;SAVE R0
5977  041152  017600  000002        MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
5978  041156  122737  000001  001246  CMPB  #APTENV,$ENV    ;;RUNNING IN APT MODE
5979  041164  001011                BNE     62$             ;;NO,GO CHECK FOR APT CONSOLE
5980  041166  132737  000100  001247  BITB  #APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
5981  041174  001405                BEQ     62$             ;;NO,GO CHECK FOR CONSOLE
5982  041176  010037  041206        MOV     R0,61$          ;;SETUP MESSAGE ADDRESS FOR APT
5983  041202  004737  041426        JSR     PC,$ATY3        ;;SPOOL MESSAGE TO APT
5984  041206  000000          61$:  .WORD   0               ;;MESSAGE ADDRESS
5985  041210  132737  000040  001247  62$:  BITB  #APTCSUP,$ENVM ;;APT CONSOLE SUPPRESSED
5986  041216  001003                BNE     60$             ;;YES,SKIP TYPE OUT
5987  041220  112046          2$:   MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
5988  041222  001005                BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
5989  041224  005726                TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
5990  041226  012600          60$:  MOV     (SP)+,R0        ;;RESTORE R0
5991  041230  062716  000002  3$:   ADD     #2,(SP)         ;;ADJUST RETURN PC
5992  041234  000002                RTI                     ;;RETURN
5993  041236  122716  000011  4$:   CMPB    #HT,(SP)        ;;BRANCH IF <HT>
5994  041242  001430                BEQ     8$
5995  041244  122716  000200        CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
5996  041250  001006                BNE     5$
5997  041252  005726                TST     (SP)+           ;;POP  <CR><LF> EQUIV
5998  041254  104401                TYPE                    ;;TYPE A CR AND LF
5999  041256  001223                $CRLF
6000  041260  105037  041414        CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
```

```
                                        BR      2$              ;;GET NEXT CHARACTER
                                5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
                                6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
                                        BNE     2$              ;;IF NO GO GET NEXT CHAR.
                                        MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
                                                                ;;AND THE NULL CHAR.
                                7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
                                        BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
                                        JSR     PC,$TYPEC       ;;GO TYPE A NULL
                                        DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
                                        BR      7$              ;;LOOP

                                ;HORIZONTAL TAB PROCESSOR

                                8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
                                9$:     JSR     PC,$TYPEC       ;;TYPE A SPACE
                                        BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
                                        BNE     9$              ;;TAB STOP
                                        TST     (SP)+           ;;POP SPACE OFF STACK
                                        BR      2$              ;;GET NEXT CHARACTER
                                $TYPEC: TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
                                        BPL     $TYPEC
                                        MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
                                        CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
                                        BNE     1$              ;;BRANCH IF NO
                                        CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
                                        BR      $TYPEX          ;;EXIT
                                1$:     CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
                                        BEQ     $TYPEX          ;;BRANCH IF YES
                                        INCB    (PC)+           ;;COUNT THE CHARACTER
                                $CHARCNT:.WORD  0               ;;CHARACTER COUNT STORAGE
                                $TYPEX: RTS     PC

                                .SBTTL  APT COMMUNICATIONS ROUTINE

                                ;;**************************************************************
                                $ATY1:  MOVB    #1,$FFLG        ;;TO REPORT FATAL ERROR
                                $ATY3:  MOVB    #1,$MFLG        ;;TO TYPE A MESSAGE
                                        BR      $ATYC
                                $ATY4:  MOVB    #1,$FFLG        ;;TO ONLY REPORT FATAL ERROR
                                $ATYC:
                                        MOV     R0,-(SP)        ;;PUSH R0 ON STACK
                                        MOV     R1,-(SP)        ;;PUSH R1 ON STACK
                                        TSTB    $MFLG           ;;SHOULD TYPE A MESSAGE?
                                        BEQ     5$              ;;IF NOT:  BR
                                        CMPB    #APTENV,$ENV    ;;OPERATING UNDER APT?
                                        BNE     3$              ;;IF NOT:  BR
                                        BITB    #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
                                        BEQ     3$              ;;IF NOT:  BR
                                        MOV     @4(SP),R0       ;;GET MESSAGE ADDR.
                                        ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
                                1$:     TST     $MSGTYPE        ;;SEE IF DONE W/ LAST XMISSION?
                                        BNE     1$              ;;IF NOT:  WAIT
                                        MOV     R0,$MSGAD       ;;PUT ADDR IN MAILBOX
                                2$:     TSTB    (R0)+           ;;FIND END OF MESSAGE
                                        BNE     2$
```

```
6057  041526  163700  001242          SUB     $MSGAD,R0       ;;SUB START OF MESSAGE
6058  041532  006200                  ASR     R0              ;;GET MESSAGE LNGTH IN WORDS
6059  041534  010037  001244          MOV     R0,$MSGLGT      ;;PUT LENGTH IN MAILBOX
6060  041540  012737  000004  001226  MOV     #4,$MSGTYPE     ;;TELL APT TO TAKE MSG.
6061  041546  000413                  BR      5$
6062  041550  017637  000004  041574  3$:     MOV     @4(SP),4$       ;;PUT MSG ADDR IN JSR LINKAGE
6063  041556  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDRESS
6064  041564  013746  177776          MOV     177776,-(SP)    ;;PUSH 177776 ON STACK
6065  041570  004737  041136          JSR     PC,$TYPE        ;;CALL TYPE MACRO
6066  041574  000000          4$:     .WORD   0
6067  041576                  5$:
6068  041576  105737  041664  10$:    TSTB    $FFLG           ;;SHOULD REPORT FATAL ERROR?
6069  041602  001416                  BEQ     12$             ;;IF NOT:  BR
6070  041604  005737  001246          TST     $ENV            ;;RUNNING UNDER APT?
6071  041610  001413                  BEQ     12$             ;;IF NOT:  BR
6072  041612  005737  001226  11$:    TST     $MSGTYPE        ;;FINISHED LAST MESSAGE?
6073  041616  001375                  BNE     11$             ;;IF NOT:  WAIT
6074  041620  017637  000004  001230  MOV     @4(SP),$FATAL   ;;GET ERROR #
6075  041626  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
6076  041634  005237  001226          INC     $MSGTYPE        ;;TELL APT TO TAKE ERROR
6077  041640  105037  041664  12$:    CLRB    $FFLG           ;;CLEAR FATAL FLAG
6078  041644  105037  041663          CLRB    $LFLG           ;;CLEAR LOG FLAG
6079  041650  105037  041662          CLRB    $MFLG           ;;CLEAR MESSAGE FLAG
6080  041654  012601                  MOV     (SP)+,R1        ;;POP STACK INTO R1
6081  041656  012600                  MOV     (SP)+,R0        ;;POP STACK INTO R0
6082  041660  000207                  RTS     PC              ;;RETURN
6083  041662     000  $MFLG:  .BYTE   0               ;;MESSG. FLAG
6084  041663     000  $LFLG:  .BYTE   0               ;;LOG FLAG
6085  041664     000  $FFLG:  .BYTE   0               ;;FATAL FLAG
6086          041666          .EVEN
6087          000200  APTSIZE=200
6088          000001  APTENV=001
6089          000100  APTSPOOL=100
6090          000040  APTCSUP=040
6091                   .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
6092
6093                   ;;*****************************************************************
6094                   ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
6095                   ;*OCTAL (ASCII) NUMBER AND TYPE IT.
6096                   ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
6097                   ;*CALL:
6098                   ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
6099                   ;*      TYPOS                   ;;CALL FOR TYPEOUT
6100                   ;*      .BYTE   N               ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
6101                   ;*      .BYTE   M               ;;M=1 OR 0
6102                   ;*                                      ;;1=TYPE LEADING ZEROS
6103                   ;*                                      ;;0=SUPPRESS LEADING ZEROS
6104                   ;*
6105                   ;*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
6106                   ;*$TYPOS OR $TYPOC
6107                   ;*CALL:
6108                   ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
6109                   ;*      TYPON                   ;;CALL FOR TYPEOUT
6110                   ;*
6111                   ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
6112                   ;*CALL:
```

```
                                        ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                                        ;*      TYPOC                   ;;CALL FOR TYPEOUT

                                STYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
                        042111          MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
                                        MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
                                        ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
                                        BR      $TYPON
                        042111  STYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
                        042113          MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
                        042110  STYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
                                        MOV     R3,-(SP)        ;;SAVE R3
                                        MOV     R4,-(SP)        ;;SAVE R4
                                        MOV     R5,-(SP)        ;;SAVE R5
                        042113          MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
                                        NEG     R4
                                        ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
                        042112          MOVB    R4,$OMODE       ;;SAVE IT FOR USE
                        042111          MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
                        000012          MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
                                        CLR     R3              ;;CLEAR THE OUTPUT WORD
                                1$:     ROL     R5              ;;ROTATE MSB INTO "C"
                                        BR      3$              ;;GO DO MSB
                                2$:     ROL     R5              ;;FORM THIS DIGIT
                                        ROL     R5
                                        ROL     R5
                                        MOV     R5,R3
                                3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
                        042112          DECB    $OMODE          ;;TYPE THIS DIGIT?
                                        BPL     7$              ;;BR IF NO
                        177770          BIC     #177770,R3      ;;GET RID OF JUNK
                                        BNE     4$              ;;TEST FOR 0
                                        TST     R4              ;;SUPPRESS THIS 0?
                                        BEQ     5$              ;;BR IF YES
                                4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
                        000060          BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
                        000040  5$:     BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
                        042106          MOVB    R3,8$           ;;SAVE FOR TYPING
                        042106          TYPE    ,8$             ;;GO TYPE THIS DIGIT
                        042110  7$:     DECB    $OCNT           ;;COUNT BY 1
                                        BGT     2$              ;;BR IF MORE TO DO
                                        BLT     6$              ;;BR IF DONE
                                        INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
                                        BR      2$              ;;GO DO THE LAST DIGIT
                                6$:     MOV     (SP)+,R5        ;;RESTORE R5
                                        MOV     (SP)+,R4        ;;RESTORE R4
                                        MOV     (SP)+,R3        ;;RESTORE R3
                        000002  000004  MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
                                        MOV     (SP)+,(SP)
                                        RTI                     ;;RETURN
                                8$:     .BYTE   0               ;;STORAGE FOR ASCII DIGIT
                                        .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
                                $OCNT:  .BYTE   0               ;;OCTAL DIGIT COUNTER
                                $OFILL: .BYTE   0               ;;ZERO FILL SWITCH
                                $OMODE: .WORD   0               ;;NUMBER OF DIGITS TO TYPE
                                .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
```

```
                                ;;**********************************************************
                                ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
                                ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
                                ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
                                ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
                                ;*REPLACED WITH SPACES.
                                ;*CALL:
                                ;*      MOV     NUM,-(SP)       ;;PUT THE BINARY NUMBER ON THE STACK
                                ;*      TYPDS                   ;;GO TO THE ROUTINE
                                $TYPDS:
042114 010046                           MOV     R0,-(SP)        ;;PUSH R0 ON STACK
042116 010146                           MOV     R1,-(SP)        ;;PUSH R1 ON STACK
042120 010246                           MOV     R2,-(SP)        ;;PUSH R2 ON STACK
042122 010346                           MOV     R3,-(SP)        ;;PUSH R3 ON STACK
042124 010546                           MOV     R5,-(SP)        ;;PUSH R5 ON STACK
042126 012746 020200                    MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
042132 016605 000020                    MOV     20(SP),R5       ;;GET THE INPUT NUMBER
042136 100004                           BPL     1$              ;;BR IF INPUT IS POS.
042140 005405                           NEG     R5              ;;MAKE THE BINARY NUMBER POS.
042142 112766 000055 000001             MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
042150 005000                   1$:     CLR     R0              ;;ZERO THE CONSTANTS INDEX
042152 012703 042330                    MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
042156 112723 000040                    MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
042162 005002                   2$:     CLR     R2              ;;CLEAR THE BCD NUMBER
042164 016001 042320                    MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
042170 160105                   3$:     SUB     R1,R5           ;;FORM THIS BCD DIGIT
042172 002402                           BLT     4$              ;;BR IF DONE
042174 005202                           INC     R2              ;;INCREASE THE BCD DIGIT BY 1
042176 000774                           BR      3$
042200 060105                   4$:     ADD     R1,R5           ;;ADD BACK THE CONSTANT
042202 005702                           TST     R2              ;;CHECK IF BCD DIGIT=0
042204 001002                           BNE     5$              ;;FALL THROUGH IF 0
042206 105716                           TSTB    (SP)            ;;STILL DOING LEADING 0'S?
042210 100407                           BMI     7$              ;;BR IF YES
042212 106316                   5$:     ASLB    (SP)            ;;MSD?
042214 103003                           BCC     6$              ;;BR IF NO
042216 116663 000001 177777             MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
042224 052702 000060            6$:     BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
042230 052702 000040            7$:     BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
042234 110223                           MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
042236 005720                           TST     (R0)+           ;;JUST INCREMENTING
042240 020027 000010                    CMP     R0,#10          ;;CHECK THE TABLE INDEX
042244 002746                           BLT     2$              ;;GO DO THE NEXT DIGIT
042246 003002                           BGT     8$              ;;GO TO EXIT
042250 010502                           MOV     R5,R2           ;;GET THE LSD
042252 000764                           BR      6$              ;;GO CHANGE TO ASCII
042254 105726                   8$:     TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
042256 100003                           BPL     9$              ;;BR IF NO
042260 116663 177777 177776             MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
042266 105013                   9$:     CLRB    (R3)            ;;SET THE TERMINATOR
042270 012605                           MOV     (SP)+,R5        ;;POP STACK INTO R5
042272 012603                           MOV     (SP)+,R3        ;;POP STACK INTO R3
042274 012602                           MOV     (SP)+,R2        ;;POP STACK INTO R2
042276 012601                           MOV     (SP)+,R1        ;;POP STACK INTO R1
```

```
6225  042300  012600                        MOV     (SP)+,R0        ;;POP STACK INTO R0
6226  042302  104401  042330                TYPE    $DBLK           ;;NOW TYPE THE NUMBER
6227  042306  016666  000002  000004        MOV     2(SP),4(SP)     ;;ADJUST THE STACK
6228  042314  012616                        MOV     (SP)+,(SP)
6229  042316  000002                        RTI                     ;;RETURN TO USER
6230  042320  023420                $DTBL:  10000.
6231  042322  001750                        1000.
6232  042324  000144                        100.
6233  042326  000012                        10.
6234  042330  000004                $DBLK:  .BLKW   4
6235                                .SBTTL  TRAP DECODER
6236
6237                                ;;**************************************************************
6238                                ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
6239                                ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
6240                                ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
6241                                ;*GO TO THAT ROUTINE.
6242
6243  042340  010046                $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
6244  042342  016600  000002                MOV     2(SP),R0        ;;GET TRAP ADDRESS
6245  042346  005740                        TST     -(R0)           ;;BACKUP BY 2
6246  042350  111000                        MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
6247  042352  006300                        ASL     R0              ;;POSITION FOR INDEXING
6248  042354  016000  042374                MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
6249  042360  000200                        RTS     R0              ;;GO TO ROUTINE
6250
6251
6252                                ;;THIS IS USE TO HANDLE THE "GETPRI" MACRO
6253
6254  042362  011646                $TRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
6255  042364  016666  000004  000002        MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
6256  042372  000002                        RTI                     ;;RESTORE THE PSW
6257
6258                                .SBTTL  TRAP TABLE
6259
6260                                ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
6261                                ;*BY THE "TRAP" INSTRUCTION.
6262
6263                                ;       ROUTINE
6264                                ;       -------
6265  042374  042362                $TRPAD: .WORD   $TRAP2
6266  042376  041136                        $TYPE   ;;CALL=TYPE     TRAP+1(104401)  TTY TYPEOUT ROUTINE
6267  042400  041712                        $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
6268  042402  041666                        $TYPOS  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
6269  042404  041726                        $TYPON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
6270  042406  042114                        $TYPDS  ;;CALL=TYPDS    TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
6271
6272
6273  042410  041042                        $SAVREG ;;CALL=SAVREG   TRAP+6(104406)  SAVE R0-R5 ROUTINE
6274  042412  041100                        $RESREG ;;CALL=RESREG   TRAP+7(104407)  RESTORE R0-R5 ROUTINE
6275  042414  044120                        TBITOFF ;;CALL=TBITO    TRAP+10(104410) TURN OFF T-BIT TRAPPING
6276  042416  044146                        TBITRESTORE     ;;CALL=TBITR    TRAP+11(104411) RETURN T-BIT TO ITS PREVIOUS CON
6277
6278                                .SBTTL  POWER DOWN AND UP ROUTINES
6279
6280                                ;;**************************************************************
```

```
6281                                    ;POWER DOWN ROUTINE
6282  042420  012737  042612  000024  $PWRDN: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
6283  042426  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
6284  042434  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
6285  042436  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
6286  042440  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
6287  042442  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
6288  042444  010446                          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
6289  042446  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
6290  042450  017746  136464                  MOV     @SWR,-(SP)      ;;PUSH @SWR ON STACK
6291  042454  010637  042616                  MOV     SP,$SAVR6       ;;SAVE SP
6292  042460  012737  042472  000024          MOV     #$PWRUP,@#PWRVEC ;;SET UP VECTOR
6293  042466  000000                          HALT
6294  042470  000776                          BR      .-2             ;;HANG UP
6295
6296                                    ;;*****************************************************************
6297                                    ;POWER UP ROUTINE
6298  042472  012737  042612  000024  $PWRUP: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
6299  042500  013706  042616                  MOV     $SAVR6,SP       ;;GET SP
6300  042504  005037  042616                  CLR     $SAVR6          ;;WAIT LOOP FOR THE TTY
6301  042510  005237  042616          1$:     INC     $SAVR6          ;;WAIT FOR THE INC
6302  042514  001375                          BNE     1$              ;;OF  WORD
6303  042516  105737  001450                  TSTB    FORTY           ;EXECUTING ON AN 11/40?
6304  042522  001003                          BNE     2$              ;BRANCH IF YES
6305  042524  011600                          MOV     (SP),R0         ;GET OLD SW. REG. CONTENTS
6306  042526  076600                          MED                     ;WRITE THE CONSOLE SW. REG.
6307  042530  000226                          WCNSSW                  ;  WITH ITS PREVIOUS VALUE
6308  042532                          2$:
6309  042532  012677  136402                  MOV     (SP)+,@SWR      ;;POP STACK INTO @SWR
6310  042536  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
6311  042540  012604                          MOV     (SP)+,R4        ;;POP STACK INTO R4
6312  042542  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
6313  042544  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
6314  042546  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
6315  042550  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
6316  042552  012737  042420  000024          MOV     #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
6317  042560  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
6318  042566  104401                          TYPE                    ;;REPORT THE POWER FAILURE
6319  042570  042620                  $PWRMG: .WORD   PWRMSG          ;;POWER FAIL MESSAGE POINTER
6320  042572  012716                          MOV     (PC)+,(SP)      ;;RESTART AT START
6321  042574  020066                  $PWRAD: .WORD   START           ;;RESTART ADDRESS
6322  042576  042766  000020  000002          BIC     #20,2(SP)       ;;CLEAR "T" BIT
6323  042604  005037  041034                  CLR     $TBIT           ;;CLEAR THE "T" BIT FLAG
6324  042610  000002                          RTI
6325  042612  000000                  $ILLUP: HALT                    ;;THE POWER UP SEQUENCE WAS STARTED
6326  042614  000776                          BR      .-2             ;;  BEFORE THE POWER DOWN WAS COMPLETE
6327  042616  000000                  $SAVR6: 0                       ;;PUT THE SP HERE
6328  042620  006412  047520  042527  PWRMSG: .ASCIZ  <12><15>?POWER FAILURE, RESTARTING PROGRAM?
6329  042626  020122  040506  046111
6330  042634  051125  026105  051040
6331  042642  051505  040524  052122
6332  042650  047111  020107  051120
6333  042656  043517  040522  000115
6334                                    .EVEN
6335
6336                                    .SBTTL  DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE
```

```
                                ;;**********************************************************************
                                ;*THIS ROUTINE WILL CONVERT A 32-BIT UNSIGNED BINARY NUMBER TO AN
                                ;*UNSIGNED OCTAL ASCIZ NUMBER.
                                ;*CALL
                                ;*      MOV     #PNTR,-(SP)     ;;POINTER TO LOW WORD OF BINARY NUMBER
                                ;*      JSR     PC,@#$DB20      ;;CALL THE ROUTINE
                                ;*      RETURN                  ;;THE ADDRESS OF THE FIRST ASCIZ CHAR. IS ON THE STACK

042664  104406                  $DB20:  SAVREG                  ;;SAVE ALL REGISTERS
042666  016601  000002                  MOV     2(SP),R1        ;;PICKUP THE POINTER TO LOW WORD
042672  012705  043003                  MOV     #$OCTVL+13.,R5  ;;POINTER TO DATA TABLE
042676  012704  000014                  MOV     #12.,R4         ;;DO ELEVEN CHARACTERS
042702  012703  177770                  MOV     #^C7,R3         ;;MASK
042706  012100                          MOV     (R1)+,R0        ;;LOWER WORD
042710  012101                          MOV     (R1)+,R1        ;;HIGH WORD
042712  005002                          CLR     R2              ;;TERMINATOR
042714  110245                  1$:     MOVB    R2,-(R5)        ;;PUT CHARACTER IN DATA TABLE
042716  010002                          MOV     R0,R2           ;;GET THIS DIGIT
042720  005304                          DEC     R4              ;;COUNT THIS CHARACTER
042722  003007                          BGT     3$              ;;BR IF NOT THE LAST DIGIT
042724  001405                          BEQ     2$              ;;BR IF IT IS THE LAST DIGIT
042726  005205                          INC     R5              ;;ALL DIGITS DONE-ADJUST POINTER FOR FIRST
042730  010566  000002                  MOV     R5,2(SP)        ;;ASCIZ CHAR. & PUT IT ON THE STACK
042734  104407                          RESREG                  ;;RESTORE ALL REGISTERS
042736  000207                          RTS     PC              ;;RETURN TO USER
042740  006203                  2$:     ASR     R3              ;;POSITION THE MASK FOR THE LAST DIGIT
042742  006001                  3$:     ROR     R1              ;;POSITION THE BINARY NUMBER FOR
042744  006000                          ROR     R0              ;;      THE NEXT OCTAL DIGIT
042746  006001                          ROR     R1
042750  006000                          ROR     R0
042752  006001                          ROR     R1
042754  006000                          ROR     R0
042756  040302                          BIC     R3,R2           ;;MASK OUT ALL JUNK
042760  062702  000060                  ADD     #'0,R2          ;;MAKE THIS CHAR. ASCII
042764  000753                          BR      1$              ;;GO PUT IT IN THE DATA TABLE
042766  000016                  $OCTVL: .BLKB   14.             ;;RESERVE DATA TABLE

                                .SBTTL  SCOPE HANDLER ROUTINE

                                ;;**********************************************************************
                                ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
                                ;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
                                ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
                                ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
                                ;*SW14=1         LOOP ON TEST
                                ;*SW11=1         INHIBIT ITERATIONS
                                ;*SW09=1         LOOP ON ERROR
                                ;*SW08=1         LOOP ON TEST IN SWR<6:0>
                                ;*CALL
                                ;*      SCOPE           ;;SCOPE=IOT

043004                          $SCOPE:
043004  005037  001444                  CLR     RETRY           ;CLEAR THE RETRY FLAG BEFORE THIS TEST
043010  005037  001432                  CLR     ERRCNT          ;CLEAR THE MULTIPLE ERROR COUNTER
```

```
 6393  043014  032777  040000  136116  1$:     BIT     #BIT14,@SWR     ;;LOOP ON PRESENT TEST?
 6394  043022  001117                          BNE     $OVER           ;;YES IF SW14=1
 6395                                  ;#####START OF CODE FOR THE XOR TESTER#####
 6396  043024  000416                  $XTSTR: BR      6$              ;;IF RUNNING ON THE "XOR" TESTER CHANGE
 6397                                                                  ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
 6398  043026  013746  000004                  MOV     @#ERRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
 6399  043032  012737  043052  000004          MOV     #5$,@#ERRVEC    ;;SET FOR TIMEOUT
 6400  043040  005737  177060                  TST     @#177060        ;;TIME OUT ON XOR?
 6401  043044  012637  000004                  MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
 6402  043050  000466                          BR      $SVLAD          ;;GO TO THE NEXT TEST
 6403  043052  022626                  5$:     CMP     (SP)+,(SP)+     ;;CLEAR THE STACK AFTER A TIME OUT
 6404  043054  012637  000004                  MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
 6405  043060  000426                          BR      7$              ;;LOOP ON THE PRESENT TEST
 6406  043062                          6$:;#####END OF CODE FOR THE XOR TESTER#####
 6407  043062  032777  000400  136050          BIT     #BIT08,@SWR     ;;LOOP ON SPEC. TEST?
 6408  043070  001407                          BEQ     2$              ;;BR IF NO
 6409  043072  017746  136042                  MOV     @SWR,-(SP)      ;;SET DESIRED TEST NUM. FROM SWR
 6410  043076  042716  000200                  BIC     #$SWRMK,(SP)    ;;STRIP AWAY UNDESIRED BITS
 6411  043102  122637  001102                  CMPB    (SP)+,$TSTNM    ;;ON THE RIGHT TEST?
 6412  043106  001465                          BEQ     $OVER           ;;BR IF YES
 6413  043110  105737  001103          2$:     TSTB    $ERFLG          ;;HAS AN ERROR OCCURRED?
 6414  043114  001421                          BEQ     3$              ;;BR IF NO
 6415  043116  123737  001115  001103          CMPB    $ERMAX,$ERFLG   ;;MAX. ERRORS FOR THIS TEST OCCURRED?
 6416  043124  101015                          BHI     3$              ;;BR IF NO
 6417  043126  032777  001000  136004          BIT     #BIT09,@SWR     ;;LOOP ON ERROR?
 6418  043134  001404                          BEQ     4$              ;;BR IF NO
 6419  043136  013737  001110  001106  7$:     MOV     $LPERR,$LPADR   ;;SET LOOP ADDRESS TO LAST SCOPE
 6420  043144  000446                          BR      $OVER
 6421  043146  105037  001103          4$:     CLRB    $ERFLG          ;;ZERO THE ERROR FLAG
 6422  043152  005037  001212                  CLR     $TIMES          ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
 6423  043156  000415                          BR      1$              ;;ESCAPE TO THE NEXT TEST
 6424  043160  032777  004000  135752  3$:     BIT     #BIT11,@SWR     ;;INHIBIT ITERATIONS?
 6425  043166  001011                          BNE     1$              ;;BR IF YES
 6426  043170  005737  001234                  TST     $PASS           ;;IF FIRST PASS OF PROGRAM
 6427  043174  001406                          BEQ     1$              ;;      INHIBIT ITERATIONS
 6428  043176  005237  001104                  INC     $ICNT           ;;INCREMENT ITERATION COUNT
 6429  043202  023737  001212  001104          CMP     $TIMES,$ICNT    ;;CHECK THE NUMBER OF ITERATIONS MADE
 6430  043210  002024                          BGE     $OVER           ;;BR IF MORE ITERATION REQUIRED
 6431  043212  012737  000001  001104  1$:     MOV     #1,$ICNT        ;;REINITIALIZE THE ITERATION COUNTER
 6432  043220  013737  043276  001212          MOV     $MXCNT,$TIMES   ;;SET NUMBER OF ITERATIONS TO DO
 6433  043226  105237  001102          $SVLAD: INCB    $TSTNM          ;;COUNT TEST NUMBERS
 6434  043232  113737  001102  001232          MOVB    $TSTNM,$TESTN   ;;SET TEST NUMBER IN APT MAILBOX
 6435  043240  011637  001106                  MOV     (SP),$LPADR     ;;SAVE SCOPE LOOP ADDRESS
 6436  043244  011637  001110                  MOV     (SP),$LPERR     ;;SAVE ERROR LOOP ADDRESS
 6437  043250  005037  001214                  CLR     $ESCAPE         ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
 6438  043254  112737  000001  001115          MOVB    #1,$ERMAX       ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
 6439  043262  013777  001102  135652  $OVER:  MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
 6440  043270  013716  001106                  MOV     $LPADR,(SP)     ;;FUDGE RETURN ADDRESS
 6441  043274  000002                          RTI                     ;;FIXES PS
 6442  043276  000310                  $MXCNT: 200.                    ;;MAX. NUMBER OF ITERATIONS
 6443                                  .SBTTL  ERROR HANDLER ROUTINE
 6444
 6445                                  ;;**************************************************************
 6446                                  ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
 6447                                  ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
 6448                                  ;*AND GO TO ERTYPE ON ERROR
```

```
                                       ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
                                       ;*SW15=1        HALT ON ERROR
                                       ;*SW13=1        INHIBIT ERROR TYPEOUTS
                                       ;*SW10=1        BELL ON ERROR
                                       ;*SW09=1        LOOP ON ERROR
                                       ;*CALL
                                       ;*      ERROR   N       ;;ERROR=EMT AND N=ERROR ITEM NUMBER

043300                                 $ERROR:
043300  113737  001102  001414                 MOVB    $TSTNM,TESTNO   ;SAVE TEST NUMBER FOR ERROR TYPE OUT
043306  005237  001432                         INC     ERRCNT          ;KEEP COUNT OF MULTIPLE ERRORS
043312  010037  001162                         MOV     R0,$REG0        ;SAVE R0
043316  010137  001164                         MOV     R1,$REG1        ;SAVE R1
043322  010237  001166                         MOV     R2,$REG2        ;SAVE R2
043326  010337  001170                         MOV     R3,$REG3        ;SAVE R3
043332  010437  001172                         MOV     R4,$REG4        ;SAVE R4
043336  010537  001174                         MOV     R5,$REG5        ;SAVE R5
043342  105237  001103                 7$:     INCB    $ERFLG          ;;SET THE ERROR FLAG
043346  001775                               BEQ     7$              ;;DON'T LET THE FLAG GO TO ZERO
043350  013777  001102  135564               MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
043356  032777  002000  135554               BIT     #BIT10,@SWR     ;;BELL ON ERROR?
043364  001402                               BEQ     1$              ;;NO - SKIP
043366  104401  001216                       TYPE    ,$BELL          ;;RING BELL
043372  005237  001112                 1$:     INC     $ERTTL          ;;COUNT THE NUMBER OF ERRORS
043376  011637  001116                       MOV     (SP),$ERRPC     ;;GET ADDRESS OF ERROR INSTRUCTION
043402  162737  000002  001116               SUB     #2,$ERRPC
043410  117737  135502  001114               MOVB    @$ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
043416  032777  020000  135514               BIT     #BIT13,@SWR     ;;SKIP TYPEOUT IF SET
043424  001004                               BNE     20$             ;;SKIP TYPEOUTS
043426  004737  043576                       JSR     PC,ERTYPE       ;;GO TO USER ERROR ROUTINE
043432  104401  001223                       TYPE    ,$CRLF
043436                                 20$:
043436  122737  000001  001246               CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
043444  001007                               BNE     2$              ;;NO,SKIP APT ERROR REPORT
043446  113737  001114  043460               MOVB    $ITEMB,21$      ;;SET ITEM NUMBER AS ERROR NUMBER
043454  004737  041436                       JSR     PC,$ATY4        ;;REPORT FATAL ERROR TO APT
043460     000                         21$:  .BYTE   0
043461     000                               .BYTE   0
043462  000777                         22$:  BR      22$             ;;APT ERROR LOOP
043464  005777  135450                 2$:   TST     @SWR            ;;HALT ON ERROR
043470  100001                               BPL     3$              ;;SKIP IF CONTINUE
043472  000000                               HALT                    ;;HALT ON ERROR!
043474  032777  001000  135436         3$:   BIT     #BIT09,@SWR     ;;LOOP ON ERROR SWITCH SET?
043502  001402                               BEQ     4$              ;;BR IF NO
043504  013716  001110                       MOV     $LPERR,(SP)     ;;FUDGE RETURN FOR LOOPING
043510  005737  001214                 4$:   TST     $ESCAPE         ;;CHECK FOR AN ESCAPE ADDRESS
043514  001402                               BEQ     5$              ;;BR IF NONE
043516  013716  001214                       MOV     $ESCAPE,(SP)    ;;FUDGE RETURN ADDRESS FOR ESCAPE
043522                                 5$:
043522  022737  040762  000042               CMP     #$ENDAD,@#42    ;;ACT-11 AUTO-ACCEPT?
043530  001001                               BNE     6$              ;;BRANCH IF NO
043532  000000                               HALT                    ;;YES
043534                                 6$:
043534  032737  001000  001140               BIT     #SW9,SWR        ;IS THE LOOP ON ERROR SWITCH UP?
043542  001414                               BEQ     EREXIT          ;BRANCH IF NOT LOOPING ON ERROR
043544  042737  177776  177572               BIC     #177776,MMR0    ;CLEAR MEMORY MANAGEMENT REG 0
```

```
                                                                        ;LEAVE BIT0 UNCHANGED
043552  012737  177777  015164            MOV     #-1,CPFLAG            ;RE-INITIALIZE CP TRAP FLAG
043560  012737  177777  015254            MOV     #-1,PAFLAG            ;RE-INITIALIZE PARITY TRAP FLAG
043566  012737  177777  015434            MOV     #-1,MMFLAG            ;RE-INITIALIZE MEMORY MANAGE. TRAP FLAG
043574  000002                    EREXIT: RTI                           ;RETURN TO TEST AFTER ERROR


                                .SBTTL  ERROR MESSAGE TYPE OUT ROUTINE


                                ;*      THIS SUBROUTINE IS CALLED BY THE ERROR HANDLER TO TYPE
                                ;*      THE ERROR MESSAGES.  IT PICKS UP THE ITEM BYTE ($ITEMB) NUMBER
                                ;*      AND USES THAT TO INDEX THROUGH THE ERROR TABLE.  THE ERROR
                                ;*      TABLE STARTS AT "$ERRTB" AND HAS FOUR (4) POINTERS FOR EACH
                                ;*      ENTRY, 'EM'  'DH'  'DT'  'DF'.  THE 'EM' POINTS TO THE ERROR
                                ;*      MESSAGE WHICH IS AN ASCIZ STRING.  THE 'DH' POINTS TO THE DATA
                                ;*      HEADER WHICH IS ANOTHER ASCIZ STRING.  THE 'DT' POINTS TO THE
                                ;*      DATA TABLE WHICH IS A GROUP OF WORDS CONTAINING THE ADDRESSES
                                ;*      OF THE DATA TO BY TYPED.  THE FORMAT OF THIS DATA IS
                                ;*      CONTROLLED BY THE 'DF' WHICH IS THE POINTER TO THE DATA FORMAT.
                                ;*      THE DATA FORMAT IS A GROUP OF BYTES WHICH CONTAIN NUMBERS
                                ;*      THAT CORRESPOND TO DIFFERENT TYPING FORMATS.
                                ;*      0       -16 BIT OCTAL FORMAT
                                ;*      1       -DECIMAL FORMAT
                                ;*      2       -18 BIT OCTAL FORMAT.  DATA IS A P.A.R. AND THE
                                ;*               OUTPUT IS THE BASE ADDRESS THAT THE P.A.R. POINTS TO.
043576  010046                    ERTYPE: MOV     R0,-(KSP)             ;SAVE R0 ON STACK
043600  005000                            CLR     R0                    ;CLEAR R0
043602  113700  001114                    MOVB    @#$ITEMB,R0           ;PUT ITEM NUMBER IN R0
043606  001004                            BNE     1$                    ;BRANCH IF IT IS NON-ZERO
043610  013746  001116                    MOV     $ERRPC,-(KSP)         ;PUT ERROR PC ON STACK FOR TYPING
043614  104402                            TYPOC                         ;TYPE FAILING PC
043616  000534                            BR      13$                   ;GO TO RETURN
043620  104401  001223            1$:     TYPE    ,$CRLF                ;TYPE CRLF
043624  005300                            DEC     R0                    ;ADJUST ITEM NUMBER TO BE A POINTER
043626  006300                            ASL     R0                    ;LEFT SHIFT ITEM NO. 3 PLACES
043630  006300                            ASL     R0
043632  006300                            ASL     R0
043634  100024                            BPL     22$                   ;BRANCH IF ITEM #LESS THAN 200
043636  032700  001000                    BIT     #BIT9,R0              ;SEE IF ITEM # WAS 3XX
043642  001415                            BEQ     21$                   ;BRANCH IF ITEM # WAS 2XX
                                                                        ;AND TYPE ERROR MESSAGE
043644  032737  000200  001140            BIT     #SW7,@#SWR            ;SEE IF SWITCH 7 IS UP
043652  001404                            BEQ     20$                   ;BRANCH IF SWITCH IS NOT UP
                                                                        ;AND TYPE DATA, ON MULTIPLE ERRORS
043654  062766  000004  000002            ADD     #4,2(KSP)             ;SKIP 'TYPE  ,$CRLF' IF SW 7 IS UP
                                                                        ;INHIBIT MULTIPLE ERROR TYPEOUTS
043662  000512                            BR      13$                   ;BRANCH TO EXIT
043664                            20$:
043664  042700  177000                    BIC     #177000,R0            ;CLEAR UPPER BYTE OF R0
043670  062700  002474                    ADD     #ER200+4,R0           ;POINT TO DATA TABLE ENTRY
043674  000424                            BR      5$                    ;GO TYPE DATA TABLE
043676  042700  177000            21$:    BIC     #177000,R0            ;CLEAR UPPER BYTE OF R0
043702  062700  000770                    ADD     #<ER200-$ERRTB>,R0 ;ADD DIFFERENCE BETWEEN
                                                                        ;ITEM 1 AND ITEM 201
                                ;;                      GET POINTER TO ERROR MESSAGE AND TYPE IT
```

```
                                ;;                      IF THE POINTER IS NOT ZERO
043706  062700  001500          22$:    ADD     #$ERRTB,R0      ;ADD BASE OF ERROR TABLE
043712  012037  043722                  MOV     (R0)+,2$        ;PUT MESSAGE POINTER IN TYPE STATEMENT
043716  001404                          BEQ     3$              ;BRANCH IF NO ERROR MESSAGE
043720  104401                          TYPE                    ;TYPE ERROR MESSAGE
043722  000000                  2$:     .WORD   0               ;POINTER TO ERROR MESSAGE
043724  104401  001223                  TYPE    ,$CRLF          ;TYPE CRLF
                                ;;                      GET THE POINTER TO THE DATA HEADER AND
                                ;;                      TYPE IT IF THE POINTER IS NOT ZERO
043730  012037  043740          3$:     MOV     (R0)+,4$        ;PUT HEADER POINTER IN TYPE STATEMENT
043734  001404                          BEQ     5$              ;BRANCH IF NO DATA HEADER
043736  104401                          TYPE                    ;TYPE THE DATA HEADER
043740  000000                  4$:     .WORD   0               ;POINTER TO DATA HEADER
043742  104401  001223                  TYPE    ,$CRLF          ;TYPE CRLF
                                ;;                      THIS IS THE START OF THE DATA OUTPUT IF THE
                                ;;                      DATA POINTER IS NOT ZERO.  R0 POINTS TO THE
                                ;;                      DATA FORMAT, R1 POINTS TO THE ADDRESS OF
                                ;;                      THE DATA WORDS.
043746  010146                  5$:     MOV     R1,-(KSP)       ;SAVE R1 ON THE STACK
043750  012001                          MOV     (R0)+,R1        ;PUT DATA TABLE POINTER IN R1
043752  001455                          BEQ     12$             ;BRANCH IF NO DATA TABLE
043754  012000                          MOV     (R0)+,R0        ;PICK UP DATA FORMAT POINTER
043756  105710                  6$:     TSTB    (R0)            ;IS THIS WORD OCTAL
043760  001003                          BNE     7$              ;BRANCH IF NOT 16-BIT OCTAL
                                ;;                      THIS IS 16 BIT OCTAL FORMAT (DF = 0)
043762  013146                          MOV     @(R1)+,-(KSP)   ;PUT WORD ON STACK FOR TYPING
043764  104402                          TYPOC                   ;TYPE THE WORD ON STACK AS 16 BIT OCTAL
043766  000441                          BR      11$             ;GET READY FOR NEXT WORD
043770  122710  000001          7$:     CMPB    #1,(R0)         ;IS THE WORD DECIMAL
043774  001003                          BNE     10$             ;BRANCH IF NOT DECIMAL
                                ;;                      THIS IS DECIMAL FORMAT (DF = 1)
043776  013146                          MOV     @(R1)+,-(KSP)   ;PUT WORD ON STACK FOR TYPING
044000  104405                          TYPDS                   ;TYPE THE WORD ON STACK AS DECIMAL
044002  000433                          BR      11$             ;GET READY FOR NEXT WORD
                                ;;                      THIS IS FORMAT 2.  DATA WORD IS A P.A.R.
                                ;;                      OUTPUT WILL BE 18-BIT. PAR LEFT SHIFTED 2.
044004  010246                  10$:    MOV     R2,-(KSP)       ;SAVE R2 ON THE STACK
044006  010346                          MOV     R3,-(KSP)       ;SAVE R3 ON THE STACK
044010  013103                          MOV     @(R1)+,R3       ;LOAD DATA WORD INTO R3
044012  005002                          CLR     R2              ;R2 WILL HOLD UPPER SIX BITS OF NUMBER
044014  012705  000006                  MOV     #6,R5           ;LEFT SHIFT <R2:R3> 6 PLACES
044020  000241                  23$:    CLC
044022  006303                          ASL     R3
044024  006102                          ROL     R2
044026  077504                          SOB     R5,23$
044030  010237  001374                  MOV     R2,PADRSH       ;STORE UPPER BITS OF ADDRESS
044034  010337  001372                  MOV     R3,PADRSL       ;STORE LOWER 16 BITS OF ADDRESS
044040  012746  001372                  MOV     #PADRSL,-(KSP)  ;PUT ADDRESS OF LOWER 16 BITS ON STACK
044044  004737  042664                  JSR     PC,$DB2O        ;CONVERT TWO WORDS TO ASCIZ
044050  062716  000005                  ADD     #5,(KSP)        ;I ONLY WANT 6 DIGITS
044054  012637  044062                  MOV     (KSP)+,31$      ;PUT POINTER AFTER TYPE CALL
044060  104401                          TYPE                    ;POINTER TO ASCIZ STRING FOLLOWS
044062  000000                  31$:    .WORD   0               ;POINTER TO ASCIZ STRING
044064  012603                          MOV     (KSP)+,R3       ;RESTORE R3 FROM STACK
044066  012602                          MOV     (KSP)+,R2       ;RESTORE R2 FROM STACK
044070  000400                          BR      11$             ;GET READY FOR NEXT WORD
```

```
 6617  044072  005200                  11$:    INC     R0              ;POINT TO NEXT FORMAT BYTE
 6618  044074  104401  044114                  TYPE    32$             ;TYPE TWO SPACES
 6619  044100  005711                          TST     (R1)            ;IS THERE ANOTHER WORD?
 6620  044102  001401                          BEQ     12$             ;BRANCH IF ALL DONE
 6621  044104  000724                          BR      6$              ;GO BACK FOR NEXT NUMBER
 6622  044106  012601                  12$:    MOV     (KSP)+,R1       ;RESTORE R1
 6623  044110  012600                  13$:    MOV     (KSP)+,R0       ;RESTORE R0
 6624  044112  000207                          RTS     PC              ;RETURN TO ERROR ROUTINE
 6625  044114  020040      000         32$:    .ASCIZ  ?  ?            ;TWO SPACES
 6626          044120                          .EVEN
 6627
 6628
 6629
 6630
 6631
 6632                                  .SBTTL  ************** SUBROUTINES UNIQUE TO THIS PROGRAM ***************
 6633
 6634
 6635
 6636
 6637
 6638
 6639                                  .SBTTL  TURN OFF AND SAVE THE T-BIT
 6640                                  ;*
 6641                                  ;*      THIS SUBROUTINE IS REACHED BY THE TRAP CALL "TBITO". IT IS
 6642                                  ;*      USED TO TURN OFF THE T-BIT IF IT IS ON.  THE PROCESSOR STATUS
 6643                                  ;*      IS SAVED IN "OLDPSW" SO THAT THE T-BIT CAN BE RESTORED TO ITS
 6644                                  ;*      PREVIOUS STATUS WHEN CONDITIONS WARRANT
 6645                                  ;*
 6646                                          .EQUIV  BIT4,TBIT
 6647  044120                          TBITOFF:
 6648  044120  032766  000020  000002          BIT     #TBIT,2(KSP)    ;IS THE T-BIT ON?
 6649  044126  001406                          BEQ     1$              ;BRANCH IF ITS NOT
 6650  044130  016637  000002  001442          MOV     2(KSP),OLDPSW   ;SAVE OLD PSW FOR RESTORING LATER
 6651  044136  042766  000020  000002          BIC     #TBIT,2(KSP)    ;CLEAR THE T-BIT
 6652  044144  000006                  1$:     RTT                     ;RETURN TO THE PROGRAM
 6653
 6654
 6655                                  .SBTTL  RESTORE T-BIT TO ITS PREVIOUS CONDITION
 6656                                  ;*
 6657                                  ;*      THIS SUBROUTINE IS REACHED BY THE TRAP CALL "TBITR". IT IS
 6658                                  ;*      USED TO RESTORE THE T-BIT AFTER A PARTICULAR TEST THAT
 6659                                  ;*      CANNOT BE RUN WITH THE T-BIT ON.  IT USES THE PROCESSOR
 6660                                  ;*      STATUS STORED IN "OLDPSW" BY "TBITO", REPLACES THE PS ON THE
 6661                                  ;*      STACK WITH IT AND DOES AN "RTT".
 6662                                  ;*
 6663  044146                          TBITRESTORE:
 6664  044146  013766  001442  000002          MOV     OLDPSW,2(KSP)   ;PUT OLD PSW ON THE STACK
 6665  044154  042737  000020  001442          BIC     #TBIT,OLDPSW    ;CLEAR T-BIT IN "OLDPSW" SO THAT IT
 6666                                                                  ;WON'T BE TURNED ON BY ACCIDENT
 6667  044162  000006                          RTT                     ;RETURN TO PROGRAM AND INHIBIT T-BIT
 6668                                                                  ;TRAPPING AFTER THIS INSTRUCTION
 6669
 6670
 6671                                  .SBTTL  CLEAR 8 PARS OR PDRS STARTING FROM ADDRESS IN R5
 6672                                  ;;**************************************************************************
```

```
                                ;*      SUBROUTINE CLRREG
                                ;*
                                ;*      THIS SUBROUTINE CLEARS 8 CONSECUTIVE MEMORY MANAGEMENT
                                ;*      REGISTERS STARTING WITH THE REGISTER POINTED TO BY R5.
                                ;*      IT SAVES R0 ON THE STACK AND LOADS A COUNT IN R0,
                                ;*      CLEARS THE PAR'S OR PDR'S, AND THEN RESTORES R0 BEFORE
                                ;*      RETURNING.
                                ;*      THE CALLING SEQUENCE IS:
                                ;*      MOV     #KIPAR0,R5      ;PUT ADDRESS OF FIRST KERNEL PAR IN R5
                                ;*      JSR     PC,CLRREG       ;GO CLEAR ALL KERNEL PAR'S
                                ;*
                                ;;*****************************************************************************

044164  010046                  CLRREG: MOV     R0,-(KSP)       ;SAVE R0 ON STACK
044166  012700  000010                  MOV     #10,R0          ;PUT COUNT IN R0
044172  005025                  1$:     CLR     (R5)+           ;CLEAR PAR ORPDR POINTED TO BY R5
044174  077002                          SOB     R0,1$           ;BRANCH BACK 7 DECIMAL TIMES.
044176  012600                          MOV     (KSP)+,R0       ;RESTORE R0 FROM STACK
044200  000207                          RTS     PC              ;RETURN TO TEST

                                .SBTTL  CLEANUP LOCATIONS THAT HOLD LOGICAL 'AND' AND 'OR'
                                ;;*****************************************************************************
                                ;*      SUBROUTINE CLEANUP
                                ;*
                                ;*      THIS SUBROUTINE IS USED TO INITIALIZE ALL THE LOCATIONS THAT
                                ;*      HOLD THE "LOGICAL AND" AND "LOGICAL OR" OF THE DATA AND ADDRESSES
                                ;*      THAT FAILED DURING THE EXECUTION OF A TEST.
                                ;*
                                ;;*****************************************************************************

044202                          CLEANUP:                        ;STARTING ADDRESS OF SUBROUTINE.
044202  005037  001416                  CLR     DATAOR          ;LOCATION FOR LOGICAL OR OF BAD DATA
044206  005037  001426                  CLR     ADDROR          ;LOCATION FOR LOGICAL OR OF ADDRESS
044212  005037  001422                  CLR     PATTOR          ;LOCATION FOR LOGICAL OR OF PATTERN LOADED
044216  012700  177777                  MOV     #-1,R0          ;LOAD -1 INTO R0 TO INITIALIZE LOGICAL AND LOCS
044222  010037  001420                  MOV     R0,DATAND       ;LOCATION FOR LOGICAL AND OF BAD DATA
044226  010037  001430                  MOV     R0,ADRAND       ;LOCATION FOR LOGICAL AND OF ADDRESS
044232  010037  001424                  MOV     R0,PATAND       ;LOCATION FOR LOGICAL AND OF PATTERN LOADED
044236  000207                          RTS     PC              ;RETURN TO TEST

                                .SBTTL  P.A.R. OR P.D.R. ADDRESS TIMED OUT WHEN REFERENCED
                                ;;*****************************************************************************
                                ;*
                                ;*      THIS SUBROUTINE IS USED TO LOG AND REPORT THE FACT THAT A
                                ;*      REFERENCE TO A P.A.R. OR A P.D.R. TIMED OUT ON THE UNIBUS.  IT
                                ;*      KEEPS A LOGICAL AND AND A LOGICAL OR OF EACH ADDRESS THAT TIMES
                                ;*      OUT.
                                ;;*****************************************************************************

044240                          TIMEOUT:                        ;STARTING ADDRESS OF SUBROUTINE
044240  005227                          INC     (PC)+           ;INCREMENT ONE TIME GATE
044242  177777                  TOFLAG: .WORD   -1              ;ONE TIME ENTANCE FLAG
044244  001401                          BEQ     10$             ;BRANCH IF FLAG IS NOW ZERO
044246  000000                          HALT                    ;I HAVE ENTERED THIS ROUTINE BEFORE
                                                                ;I FINISHED REPORTING THE FIRST ERROR
                                                                ;THE SECOND ENTRY ADDRESS IS ON THE
                                                                ;STACK AND THE FIRST ERROR CONDITION
```

```
                                                                ;IS PROBABLY STILL LOCKED UP .
                        012637  001436          10$:    MOV     (KSP)+,OLDPC    ;SAVE RETURN ADDRESS
                        012637  001440                  MOV     (KSP)+,OLDPS    ;SAVE OLD PSW
                        012706  001100                  MOV     #KERSTK,KSP     ;RESTORE STACK POINTER
                        050037  001426          1$:     BIS     R0,ADDROR       ;PERFORM LOGICAL OR OF FAILING ADDRESS
                        005100                          COM     R0              ;GET R0 READY FOR AND
                        040037  001430                  BIC     R0,ADRAND               ;PERFORM LOGICAL AND
                        005100                          COM     R0              ;PUT R0 BACK AS IT WAS
                        105737  001103                  TSTB    $ERFLG          ;IS THIS THE FIRST ERROR
                        001002                          BNE     2$              ;BRANCH IF NOT FIRST ERROR
                        104201                          ERROR   201             ;NO REGISTER RESPONSE.
                        000401                          BR      3$              ;BRANCH TO EXIT
                        104301                  2$:     ERROR   301             ;CONTINUE NO RESPONSE TABLE
                        012737  177777  044242  3$:     MOV     #-1,TOFLAG      ;RESET ONE TIME GATE
                        013746  001440                  MOV     OLDPS,-(KSP)    ;RESTORE OLD PSW
                        013746  001436                  MOV     OLDPC,-(KSP)    ;PUSH RETURN ADDRESS BACK ON THE STACK
                        000006                          RTT                     ;RETURN TO THE TEST

                                        .SBTTL  DUAL ADDRESSING WHEN LOADING A P.A.R. OR P.D.R.
                                        ;;*****************************************************************************
                                        ;*
                                        ;*      THIS SUBROUTINE WILL LOG AND REPORT ALL DUAL ADDRESSING ERRORS
                                        ;*      FOUND IN BOTH P.A.R.'S AND P.D.R.'S.  A "LOGICAL OR" AND A
                                        ;*      "LOGICAL AND" OF THE WRITTEN ADDRESSES WILL BE MAINTAINED IN
                                        ;*      'ADDROR' AND 'ADRAND'.  THE LOG ON THE ADDITIONAL OR FAILING,
                                        ;*      ADDRESSES WILL BE MAINTAINED IN 'DATAOR' AND 'DATAND'.
                                        ;*
                                        ;;*****************************************************************************
        044334
        044334  012637  001436  DUALADR:                        ;STARTING LOCATION OF SUBROUTINE
                        001436          MOV     (KSP)+,OLDPC    ;SAVE RETURN ADDRESS IN CASE OF LOOP
        044340  050037  001426          BIS     R0,ADDROR       ;LOGICAL OR OF WRITTEN ADDRESS
        044344  005100                  COM     R0              ;GET R0 READY FOR AND
        044346  040037  001430          BIC     R0,ADRAND               ;PERFORM LOGICAL AND
        044352  005100                  COM     R0              ;PUT R0 BACK AS IT WAS
        044354  050137  001416          BIS     R1,DATAOR       ;LOGICAL OR OF DUALED ADDRESS
        044360  005101                  COM     R1              ;GET R1 READY FOR AND
        044362  040137  001420          BIC     R1,DATAND               ;PERFORM LOGICAL AND
        044366  005101                  COM     R1              ;PUT R1 BACK AS IT WAS
        044370  105737  001103          TSTB    $ERFLG          ;SEE IF THIS IS FIRST ERROR
        044374  001002                  BNE     1$              ;BRANCH IF NOT FIRST ERROR
        044376  104202                  ERROR   202
        044400  000401                  BR      2$              ;BRANCH TO EXIT
        044402  104302          1$:     ERROR   302
        044404  000177  135026  2$:     JMP     @OLDPC          ;RETURN TO TEST


                                        .SBTTL  COUNT PATTERN ERRORS IN P.A.R.'S OR P.D.R.'S
                                        ;;*****************************************************************************
                                        ;*
                                        ;*      THIS SUBROUTINE IS USED TO LOG AND REPORT THE COUNT PATTERN
```

```
6779                                    ;*      ERRORS OCCURRING WHEN TESTING THE P.A.R.'S AND P.D.R.'S.
6780                                    ;*      THE "LOGICAL OR" AND "LOGICAL AND" OF VARIOUS DATA WILL BE
6781                                    ;*      MAINTAINED AS FOLLOWS:
6782                                    ;*      ADDRESSES OF FAILING REGISTERS IN 'ADDROR' AND 'ADRAND'
6783                                    ;*      DATA FETCHED FROM REGISTERS IN 'DATAOR' AND 'DATAND'
6784                                    ;*      PATTERN LOADED INTO THE REGISTERS IN 'PATTOR' AND 'PATAND'.
6785                                    ;*
6786                                    ;;*************************************************************************
```

```
6787  044410                          PARCOUNT:                       ;STARTING ADDRESS OF THIS SUBROUTINE
6788  044410  012637  001436                  MOV     (KSP)+,OLDPC    ;SAVE RETURN ADDRESS IN CASE OF LOOP
6789  044414  050037  001426                  BIS     R0,ADDROR       ;LOGICAL OR OF FAILING ADDRESS
6790  044420  005100                          COM     R0              ;GET R0 READY FOR AND
6791  044422  040037  001430                  BIC     R0,ADRAND               .PERFORM LOGICAL AND
6792  044426  005100                          COM     R0              ;PUT R0 BACK AS IT WAS
6793  044430  050137  001422                  BIS     R1,PATTOR       ;LOGICAL OR OF PATTERN LOADED
6794  044434  005101                          COM     R1              ;GET R1 READY FOR AND
6795  044436  040137  001424                  BIC     R1,PATAND               .PERFORM LOGICAL AND
6796  044442  005101                          COM     R1              ;PUT R1 BACK AS IT WAS
6797  044444  050237  001416                  BIS     R2,DATAOR       ;LOGICAL OR OF DATA FETCHED
6798  044450  005102                          COM     R2              ;GET R2 READY FOR AND
6799  044452  040237  001420                  BIC     R2,DATAND               .PERFORM LOGICAL AND
6800  044456  005102                          COM     R2              ;PUT R2 BACK AS IT WAS
6801  044460  105737  001103                  TSTB    $ERFLG          ;SEE IF THIS IS THE FIRST ERROR
6802  044464  001002                          BNE     1$              ;BRANCH IF NOT FIRST ERROR
6803  044466  104203                          ERROR   203
6804  044470  000401                          BR      2$              ;BRANCH TO EXIT
6805  044472  104303                  1$:     ERROR   303
6806  044474  000177  134736          2$:     JMP     @OLDPC          ;RETURN TO TEST
6807
6808
6809
6810
6811
6812          000001                  .END
```

```
ABASE = 000000            738     779
ACDW1 = 000000            738     781
ACDW2 = 000000            738     782
ACPUOP= 000000            738     753
ADDROR  001426            832#    2195    2199    2202    6705*   6733*   6759*   6789*
ADDW0 = 000000            738     783
ADDW1 = 000000            738     784
ADDW10= 000000            738     793
ADDW11= 000000            738     794
ADDW12= 000000            738     795
ADDW13= 000000            738     796
ADDW14= 000000            738     797
ADDW15= 000000            738     798
ADDW2 = 000000            738     785
ADDW3 = 000000            738     786
ADDW4 = 000000            738     787
ADDW5 = 000000            738     788
ADDW6 = 000000            738     789
ADDW7 = 000000            738     790
ADDW8 = 000000            738     791
ADDW9 = 000000            738     792
ADEVCT= 000000            738     744
ADEVM = 000000            738     780
ADRAND  001430            833#    2195    2199    2202    6709*   6735*   6761*   6791*
AENV  = 000000            738     749
AENVM = 000000            738     750
AFATAL= 000000            738     741
AMADR1= 000000            738     766
AMADR2= 000000            738     770
AMADR3= 000000            738     773
AMADR4= 000000            738     776
AMAMS1= 000000            738     760
AMAMS2= 000000            738     768
AMAMS3= 000000            738     771
AMAMS4= 000000            738     774
AMSGAD= 000000            738     746
AMSGLG= 000000            738     747
AMSGTY= 000000            738     740
AMTYP1= 000000            738     761
AMTYP2= 000000            738     769
AMTYP3= 000000            738     772
AMTYP4= 000000            738     775
APASS = 000000            738     743
APRIOR= 000000            738
APTCSU= 000040            5985    6090#
APTENV= 000001            5978    6046    6088#   6481
APTSIZ= 000200            2630    6087#
APTSPD= 000100            5980    6048    6089#
ASWREG= 000000            738     751
ATESTM= 000000            738     742
AUNIT = 000000            738     745
AUSWR = 000000            738     752
AVECT1= 000000            738     777
AVECT2= 000000            738     778
BADPC   001412            826#    2180    2181    2183    2185    2187    2395*   2434*   2479*
BIT0  = 000001            525#    4123    4375    4475    4478    4556    4881    4997    5128    5196    5250    5318    5366
```

```
BIT00 = 000001           515#   525
BIT01 = 000002           514#   524
BIT02 = 000004           513#   523
BIT03 = 000010           512#   522
BIT04 = 000020           511#   521
BIT05 = 000040           510#   520
BIT06 = 000100           509#   519
BIT07 = 000200           508#   518
BIT08 = 000400           507#   517   6407
BIT09 = 001000           506#   516   6417   6491
BIT1  = 000002           524#
BIT10 = 002000           505#  6469
BIT11 = 004000           504#  6424
BIT12 = 010000           503#  4464   4467   4943   5894
BIT13 = 020000           502#  6476
BIT14 = 040000           501#  6393
BIT15 = 100000           500#  2906
BIT2  = 000004           523#  4517   4712   4714   4719   4722   4725
BIT3  = 000010           522#
BIT4  = 000020           521#  4028   4076   4336   4338   4652   6646
BIT5  = 000040           520#
BIT6  = 000100           519#  4998
BIT7  = 000200           518#
BIT8  = 000400           517#  3812   3827   3842   3857   3872   3887   3902   3917   3932   3947   3968   3982
BIT9  = 001000           516#  6544
BPTVEC= 000014           532#
CACHVE= 000114           608#  2663*  2664*
CLEANU  044202          2667   3030   3062   3093   3125   3167   3228   3288   3349   3420   3465   3507   3550
                        6703#
CLRREG  044164           3170   3186   3231   3247   3291   3307   3352   3368   3788   3790   3792   3794   6686#
CONTRL  001352            804#  2432   2435   2575*  2652*
CPFLAG  015164           2386#  2402*  2656*  6506*
CPUER   015162           2385#  2661   2722   3039   3071   3102   3134   4070   4553   4768
CPUERR  001354            805#  2400   2715   2716   4078   4346   4516
CPUEXP  001364            813#  2659*
CR    = 000015            440#  6024   6034
CRLF  = 000200            441#   607#  1321   1335   1598   1818   1831   1840   1881   1945   1956   1979   2023
                         2031   2037   2041   2053   2160   2643   5995   6034
DATAND  001420            829#  2199   2202   6708*  6765*  6799*
DATAOR  001416            828#  2199   2202   6704*  6763*  6797*
DDISP = 177570            447#   705   2618
DF1     014660            893   1279   2277#
DF11    014706            945   2286#
DF12    014712            951   2288#
DF13    014716            957   2290#
DF14    014723            963   2292#
DF15    014727            970   2294#
DF16    014736            976   2297#
DF17    014747            982   2300#
DF2     014662            899   2278#
DF20    014754            988   2302#
DF201   015144           1291   2357#
DF202   015147           1300   2358#
DF203   015153           1308   2360#
DF21    014757            994   1012   2303#
DF22    014763           1000   2305#
```

```
DF23    014770          1006    1018    1024    2307#
DF27    014775          1030    2309#
DF3     014665           906     913    2279#
DF30    015000          1037    2310#
DF31    015005          1044    2312#
DF32    015014          1050    2315#
DF33    015020          1057    2317#
DF34    015027          1063    2320#
DF35    015032          1069    1075    2321#
DF37    015036          1081    1087    1147    1219    2323#
DF41    015043          1093    1099    1105    1111    1117    1123    2325#
DF47    015047          1134    2327#
DF5     014672           920    2281#
DF50    015056          1141    2330#
DF52    015064          1153    2332#
DF53    015070          1159    2334#
DF54    015074          1165    2336#
DF55    015100          1171    1177    1183    1189    1195    2338#
DF6     014676           927    2283#
DF62    015103          1201    2339#
DF63    015107          1207    2341#
DF64    015113          1213    2343#
DF66    015117          1225    1243    2345#
DF67    015123          1231    2347#
DF7     014703           933     939    2285#
DF70    015127          1237    2349#
DF72    015132          1249    1255    2350#
DF74    015135          1261    2351#
DF75    015137          1267    1273    2352#
DH1     007755           891    1277    1809#
DH11    010362           943    1856#
DH12    010422           949    1862#
DH13    010462           955    1868#
DH14    010533           961    1875#
DH15    010573           967    1881#
DH16    010722           974    1897#
DH17    011031           980    1909#
DH2     010000           897    1813#
DH20    011101           986    1916#
DH201   013540          1287    2156#
DH202   013567          1295    2160#
DH203   013647          1304    2169#
DH21    011131           992    1010    1921#
DH22    011171           998    1927#
DH23    011240          1004    1016    1022    1934#
DH27    011310          1028    1941#
DH3     010034           903     910    1818#
DH30    011340          1034    1945#
DH31    011440          1041    1956#
DH32    011574          1048    1973#
DH33    011634          1054    1979#
DH34    011754          1061    1993#
DH35    012004          1067    1073    1997#
DH37    012044          1079    1145    1217    2003#
DH40    012114          1085    2010#
DH41    012164          1091    1097    1103    1109    1115    1121    2017#
```

```
DH47    012224          1128    2023#
DH5     010142           917    1831#
DH50    012467          1138    2053#
DH52    012600          1151    2066#
DH53    012640          1157    2072#
DH54    012700          1163    2078#
DH55    012740          1169    1175    1181    2084#
DH6     010226           924    1840#
DH60    012770          1187    1193    2088#
DH62    013020          1199    2092#
DH63    013060          1205    2098#
DH64    013120          1211    2104#
DH66    013160          1223    2110#
DH67    013220          1229    2116#
DH7     010332           931     937    1852#
DH70    013260          1235    2122#
DH71    013310          1241    2126#
DH72    013350          1247    2132#
DH73    013421          1253    2139#
DH74    013451          1259    2144#
DH75    013470          1265    1271    2147#
DISPLA  001142           705#   2618#   2626#   2703#   2887#   3029#   3786#   4112#   4986#   5355#   6439#   6468#
DISPRE  000174           628#   2626
DSWR  = 177570           446#    704    2617
DT1     013726           892    1278    2180#
DT11    014016           944    2191#
DT12    014030           950    2193#
DT13    014042           956    2195#
DT14    014056           962    2197#
DT15    014070           969    2199#
DT16    014110           975    2202#
DT17    014134           981    2206#
DT2     013734           898    2181#
DT20    014150           987    2208#
DT201   014620          1290    2268#
DT202   014630          1299    2270#
DT203   014642          1307    2272#
DT21    014160           993    1011    2210#
DT22    014172           999    2212#
DT23    014206          1005    1017    1023    2214#
DT27    014222          1029    2216#
DT3     013744           905     912    2183#
DT30    014232          1036    2218#
DT31    014246          1043    2220#
DT32    014266          1049    2223#
DT33    014300          1056    2225#
DT34    014320          1062    2228#
DT35    014330          1068    1074    2230#
DT37    014342          1080    1086    1146    1218    2232#
DT41    014356          1092    1098    1104    1110    1116    1122    2234#
DT47    014370          1133    2236#
DT5     013760           919    2185#
DT50    014410          1140    2239#
DT52    014426          1152    2242#
DT53    014440          1158    2244#
DT54    014452          1164    2246#
```

CROSS REFERENCE TABLE -- USER SYMBOLS

```
DT55    014464          1170    1176    1182    1188    1194    2248#
DT6     013772           926    2187#
DT62    014474          1200    2250#
DT63    014506          1206    2252#
DT64    014520          1212    2254#
DT66    014532          1224    1242    2256#
DT67    014544          1230    2258#
DT7     014006           932     938    2189#
DT70    014556          1236    2260#
DT72    014566          1248    1254    2262#
DT74    014576          1260    2264#
DT75    014604          1266    1272    2265#
DUALAD  044334          3195    3256    3316    3377    6757#
EMTVEC= 000030           535#   2587*   2588*
EM1     002520           890     896    1312#
EM10    003261           936    1377#
EM11    003312           942    1382#
EM12    003354           948    1388#
EM13    003410           954    1393#
EM14    003456           960    1400#
EM15    003517           966    1406#
EM16    003561           973    1412#
EM17    003623           979    1418#
EM20    003667           985    1425#
EM201   007520          1286    1779#
EM202   007575          1294    1787#
EM203   007666          1303    1797#
EM21    003726           991    1431#
EM22    003773           997    1438#
EM23    004047          1003    1446#
EM24    004110          1009    1452#
EM25    004145          1015    1457#
EM26    004217          1021    1465#
EM27    004303          1027    1474#
EM3     002601           902    1321#
EM30    004363          1033    1483#
EM31    004432          1040    1490#
EM32    004511          1047    1498#
EM33    004572          1053    1507#
EM34    004643          1060    1514#
EM35    004712          1066    1521#
EM36    004765          1072    1529#
EM37    005042          1078    1537#
EM4     002717           909    1276    1335#
EM40    005133          1084    1547#
EM41    005220          1090    1556#
EM42    005271          1096    1563#
EM43    005341          1102    1570#
EM44    005410          1108    1577#
EM45    005455          1114    1584#
EM46    005524          1120    1591#
EM47    005575          1126    1598#
EM5     003042           916    1351#
EM50    005661          1137    1608#
EM51    005763          1144    1620#
EM52    006027          1150    1627#
```

F11

CROSS REFERENCE TABLE -- USER SYMBOLS

```
EM53    006102          1156    1635#
EM54    006153          1162    1642#
EM55    006220          1168    1649#
EM56    006261          1174    1655#
EM57    006342          1180    1664#
EM6     003115           923    1359#
EM60    006411          1186    1671#
EM61    006445          1192    1676#
EM62    006504          1198    1682#
EM63    006537          1204    1687#
EM64    006567          1210    1692#
EM65    006635          1216    1699#
EM66    006716          1222    1708#
EM67    006764          1228    1715#
EM7     003205           930    1369#
EM70    007031          1234    1722#
EM71    007106          1240    1730#
EM72    007151          1246    1736#
EM73    007233          1252    1745#
EM74    007307          1258    1753#
EM75    007377          1264    1763#
EM76    007451          1270    1771#
ENTPT2  021510           866    2884#
ENTPT3  022072           867    3026#
ENTPT4  025066           868    3783#
ENTPT5  027204           869    4109#
ENTPT6  033674           870    4983#
ENTPT7  036000           871    5352#
EREXIT  043574          6503    6509#
ERRCNT  001432           834#   3040    3042    3072    3074    3103    3105    3135    3137    3204    3206    3265    3267
                        3325    3327    3386    3388    3439    3441    3484    3486    3526    3528    3569    3571    6392*
                        6459*
ERRVEC= 000004           528#   2615    2616*   2627*   2661*   2662*   2706*   2722*   3031*   3039*   3063*   3071*   3094*
                        3102*   3126*   3134*   4011*   4014*   4070*   4343*   4346*   4392*   4396*   4553*   4654*   4667*
                        4765*   4768*   6398    6399*   6401*   6404*
ERTYPE  043576          6478    6531#
ER200   002470          1283#   6555    6558
FORTY   001450           846#   2396    2429    2567*   2572*   2650    2713    4012    4332    4458    4502    4692    4707
                        4733    5024    6303
FSTTST  001362           811#   2538*   2540*   2542*   2544*   2546*   2548*   2550*   2669
GNS     = ****** U       627    2642    5863    5870    6266    6267    6268    6269    6270    6273    6274    6275    6276
HITMIS  001356           808#   2433
HOLFLG  001434           835#   4009*   4030*   4035    4039*
HT      = 000011         438#   5993    6034
IOTVEC= 000020           533#   2585*   2586*
KERSTK= 001100           599#   2565    2668    2848    2853    4743    5397    5678    5782    6732
KERVEC  015544          2512#   4835
KIPAR0= 172340           588#   3033    3169    3171    3184    3185    3198    3423    3597    3598*   3600    3605    3608
                        3789    3800*   4118*   4876*   4992*   5081*   5361*
KIPAR1= 172342           589#   3801*   4119*   4877*   4993*   5082*   5362*
KIPAR2= 172344           590#   3802*   4120*   4878*   4994*   5083*   5363*
KIPAR3= 172346           591#   3803*   4121*   4879*   4995*   5084*   5364*
KIPAR4= 172350           592#   3807*   3822*   3837*   3852*   3867*   3882*   3897*   3912*   3927*   3942*   3963*   3977*
                        4016*   4045    4049*   4064*   4125*   4181*   4238*   4291    4314*   4329    4887*   5000*   5085*
                        5385*   5509*   5664*
KIPAR5= 172352           593#   4017*   4033    4037    4046    4050*   4052    4239*   4254    4269    4282*   4313*   4888*
```

```
                    5086*
KIPAR6= 172354       594*  4889*  5087*  5125*  5136*
KIPAR7= 172356       595*  3036   3187   3201   3432   3804*  4122*  4880*  4996*  5088*  5126*  5195*  5365*
KIPDR0= 172300       577*  3096   3290   3292   3305   3306   3319   3510   3665   3666*  3668   3673   3676
                    3787   3797   4113*  4871*  4987*  5102   5109   5117*  5121*  5134*  5141   5149*  5153*
                    5172*  5176*  5231   5239*  5263   5271*  5294*  5356*  5370
KIPDR1= 172302       578*  4114*  4872*  4988*  5118*  5150*  5173*  5240*  5272*  5295*  5357*
KIPDR2= 172304       579*  4115*  4873*  4989*  5119*  5151*  5174*  5241*  5273*  5296*  5358*
KIPDR3= 172306       580*  4116*  4874*  4990*  5359*
KIPDR4= 172310       581*  4124*  4126   4180*  4182   4235*  4318*  4328   4585*  4883*  4999*  5004   5010
                    5015*  5016   5035*  5384*  5390*  5507*  5530*  5532*  5542*  5544*  5555*  5557*  5569*
                    5571*  5583*  5585*  5598*  5600*  5614*  5616*  5663*
KIPDR5= 172312       582*  4236*  4253   4268   4273   4275*  4315*  4884*
KIPDR6= 172314       583*  4885*  5135*
KIPDR7= 172316       584*  3099   3308   3322   3519   4117*  4875*  4991*  5023   5037   5120*  5122*  5130
                    5152*  5154*  5175*  5177*  5242*  5274*  5297*  5360*
LF    = 000012       439*  6028   6034
LOOP    020612      2648*  5905
MED   = 076600       609*  2568   4334   4348   4465   4476   4509   4710   4720   4726   6306
MEMER   015252      2419*  2663
MEMERR  001360       810*  2431
MMEXP   001366       814*  2187   2484   2488   2660*  4139*  4142*  4195*  4198*  4245*  4249*  4260*  4264*
                    4285*  4295*  4319*  4323*  4325*  4788*
MMFLAG  015434      2472*  2492*  2658*  6508*
MMR0  = 177572       600*  2482   2491*  2526   2654*  2888   2962   2965*  3812*  3827*  3842*  3857*  3872*
                    3887*  3902*  3917*  3932*  3947*  3968*  3982*  4021*  4123*  4375*  4556*  4600   4609
                    4620*  4705   4767*  4847*  4881*  4926   4942*  4997*  5045*  5196*  5318*  5366*  5482
                    5484*  5628   5630*  5759   5761*  6504*
MMR1  = 177574       601*  2936
MMR2  = 177576       602*  2483   2527   2963   2972   4601   4610   4706   4925   5483   5629   5760
MMTRAP  015432      2471*  2665   4621   4828   4950   5042   5475   5622   5751
MMVEC = 000250        544*  2665*  2666*  4587*  4588*  4593*  4621*  4668*  4828*  4835*  4836*  4906*  4907*
                    4950*  5001*  5042*  5389*  5475*  5511*  5622*  5671*  5751*
NXTTST  001446        845*  2449   2704*  2754*  2814*  2845*  2882*  2933*  2960*  3024*  3060*  3091*  3123*
                    3165*  3226*  3286*  3347*  3417*  3462*  3504*  3547*  3593*  3628*  3661*  3696*  3735*
                    3776*  4007*  4106*  4177*  4233*  4310*  4372*  4579*  4650*  4786*  4867*  4979*  5075*
                    5220*  5348*  5505*  5653*  5777*  5814*
OLDPC   001436        837*  2393*  2395   2404   2427*  2434   2447*  2449*  2453   2480*  2494   2513*  2520*
                    2529   5480*  5487   5626*  5633   5757*  5764   6730*  6744   6758*  6772   6788*  6806
OLDPS   001440        839*  2394*  2403   2428*  2452   2481*  2493   2514*  2521*  2528   5481*  5486   5627*
                    5632   5758*  5763   6731*  6743
OLDPSW  001442        841*  6650*  6664   6665*
PADRSH  001374        818*  6606*
PADRSL  001372        816*  6607*  6608
PAFLAG  015254      2420*  2451*  2657*  6507*
PARCOU  044410      3430   3475   3517   3560   6787*
PARTAB  001452        857*  3741*  3747   3750
PATAND  001424        831*  2202   6710*  6795*
PATTOR  001422        830*  2202   6706*  6793*
PCONTR  001400        821*  2183   2432*
PCPUER  001410        825*  2181   2265   2400*  4026*  4028   4076*  4078*  4516*  4517
PHITMI  001402        822*  2183   2433*
PIRQ  = 177772        445*
PIRQVE= 000240        539*
PMMR0   001404        823*  2185   2187   2236   2239   2242   2246   2258   2482*  2488   2526*  4128*  4143
                    4184*  4199   4600*  4614   4705*  4729   4829   4926*  4939   5482*  5628*  5759*
```

```
PMMR2   001406          824*  2185  2187  2236  2239  2242  2244  2258  2483*  2527*  4601*  4611  4706*
                       4753   4925* 4927  5483* 5629* 5760*
PPARER  001376          820*  2183  2431* 2437
PR0   = 000000          462*
PR1   = 000040          463*
PR2   = 000100          464*
PR3   = 000140          465*
PR4   = 000200          466*
PR5   = 000240          467*
PR6   = 000300          468*
PR7   = 000340          469*
PS    = 177776          442*   443
PSW   = 177776          443*  2512  2522  2671* 2673* 2708  2709  2847* 2849* 2851* 3991* 4671* 4735*
                       4737* 4761* 4762* 4764* 4806* 4811* 4848* 5021  5022* 5101* 5198* 5223* 5320*
                       5394* 5408* 5417* 5426* 5434* 5445* 5456* 5466* 5514* 5522* 5539* 5551* 5565*
                       5579* 5593* 5608* 5662* 5673* 5686* 5695* 5704* 5712* 5722* 5733* 5743* 5752*
                       5779* 5816*
PWRMSG  042620          6319  6328*
PWRVEC= 000024           534* 2591* 2592* 6282* 6283* 6292* 6298* 6316* 6317*
RDMMR0  015612          2518  2526*
RESREG= 104407          6274* 6362
RESVEC= 000010           529* 2602* 2609*
RETRY   001444           843* 2444  2446* 6391*
RLFGIN= 000104           613* 4349  4353
RLJAM = 000100           611* 2569  4335  4339  4462  4466  4506  4510
RLSERV= 000101           612* 4473  4477
RLWHAM= 000105           614* 4711  4715
RWHAM = 000022           610* 4718  4721
SAVREG= 104406          6273* 6347
SR0   = 177572           548*  600
SR1   = 177574           549*  601
SR2   = 177576           550*  602
SR3   = 172516           551*
STACK = 001100           433*  599   606  2583  4391  4439  4534  4539  4544  4555  4688  5817  5830
START   020066          2539  2541  2543  2545  2547  2549  2565* 6321
STKEXP  001370           815* 4390* 4401* 4412* 4440  4451
STKLMT= 177774           444* 2710  2711  4398* 4438* 4446* 4488* 4526* 4533* 4536* 4538* 4541* 4543*
                        4546* 4557*
STRTAB  001462           865* 2674
STRT1   020000           631  2538*
STRT2   020010           633  2540*
STRT3   020020           635  2542*
STRT4   020030           637  2544*
STRT5   020040           639  2546*
STRT6   020050           641  2548*
STRT7   020060           643  2550*
SWR     001140           704* 2581  2617* 2619  2625* 2632* 5894  6290  6309* 6393  6407  6409  6417
                        6424  6469  6476  6488  6491  6502  6547
SWREG   000176           629* 2625
SW0   = 000001           497*
SW00  = 000001           487*  497
SW01  = 000002           486*  496
SW02  = 000004           485*  495
SW03  = 000010           484*  494
SW04  = 000020           483*  493
SW05  = 000040           482*  492
```

```
SW06   = 000100            481#    491
SW07   = 000200            480#    490
SW08   = 000400            479#    489
SW09   = 001000            478#    488
SW1    = 000002            496#
SW10   = 002000            477#
SW11   = 004000            476#
SW12   = 010000            475#
SW13   = 020000            474#
SW14   = 040000            473#
SW15   = 100000            472#
SW2    = 000004            495#
SW3    = 000010            494#
SW4    = 000020            493#
SW5    = 000040            492#
SW6    = 000100            491#
SW7    = 000200            490#    6547
SW8    = 000400            489#
SW9    = 001000            488#    6502
TBIT   = 000020           4652#    4687    6646#   6648    6651    6665
TBITO  = 104410           2756     4374    4653    5098    6275#
TBITOF   044120           6275     6647#
TBITR  = 104411           2816     4581    4981    5350    6276#
TBITRE   044146           6276     6663#
TBITVE= 000014             530#    2599*   2600*
TESTNO   001414            827#    2180    2181    2183    2185    2187    2189    2191    2193    2195    2197    2199    2202
                          2206     2208    2210    2212    2214    2216    2218    2220    2223    2225    2228    2230    2232
                          2234     2236    2239    2242    2244    2246    2248    2250    2252    2254    2256    2258    2260
                          2262     2264    2265    2268    2270    2272    6458*
TIMEOU   044240           3031     3063    3094    3126    6721#
TKVEC  = 000060            537#
TOFLAG   044242           6723#    6742*
TPVEC  = 000064            538#
TRAPVE= 000034             536#    2589*   2590*
TRTVEC= 000014             531#
TST1     020776            865     2670    2700#
TST10    022062           2960     2989    3022#
TST11    022206           3024     3041    3058#
TST12    022310           3060     3073    3089#
TST13    022412           3091     3104    3121#
TST14    022514           3123     3136    3163#
TST15    022720           3165     3205    3224#
TST16    023124           3226     3266    3284#
TST17    023330           3286     3326    3345#
TST2     021130           2704     2723    2752#
TST20    023534           3347     3387    3415#
TST21    023666           3417     3440    3460#
TST22    024020           3462     3485    3502#
TST23    024152           3504     3527    3545#
TST24    024304           3547     3570    3591#
TST25    024416           3593     3626#
TST26    024530           3628     3659#
TST27    024642           3661     3694#
TST3     021340           2754     2812#
TST30    024754           3696     3733#
TST31    025056           3735     3774#
```

```
TST32   026620            3776    4005#
TST33   027166            4007    4072    4104#
TST34   027512            4106    4175#
TST35   027712            4177    4231#
TST36   030210            4233    4308#
TST37   030454            4310    4370#
TST4    021430            2814    2843#
TST40   031562            4372    4577#
TST41   032002            4579    4648#
TST42   032640            4650    4784#
TST43   033156            4786    4834    4865#
TST44   033662            4867    4977#
TST45   034342            4979    5043    5073#
TST46   035240            5075    5218#
TST47   035766            5220    5346#
TST5    021500            2845    2854    2880#
TST50   036730            5348    5477    5503#
TST51   037616            5505    5623    5651#
TST52   040364            5653    5754    5775#
TST53   040476            5777    5813#
TST6    021666            2882    2931#
TST7    021716            2933    2958#
TYPDS = 104405            5867    5874    6270#   6593
TYPE  = 104401            2640    5861    5868    5875    5998    6151    6226    6266#   6318    6471    6479    6538    6565
                          6567    6572    6574    6612    6618
TYPOC = 104402            6267#   6536    6587
TYPON = 104404            6269#
TYPOS = 104403            6268#
UIPAR0= 177640             566#   3065    3230    3232    3245    3246    3259    3468    3632    3633*   3635    3640    3643
                          3793    4656*   4796*   4894*   5089*   5377*
UIPAR1= 177642             567#   4657*   4797*   4895*   5090*   5378*
UIPAR2= 177644             568#   4658*   4798*   4896*   5091*   5379*
UIPAR3= 177646             569#   4659*   4799*   4897*   5092*   5380*
UIPAR4= 177650             570#   4669*   4898*   5093*   5386*   5510*   5665*
UIPAR5= 177652             571#   4802*   4899*   5094*
UIPAR6= 177654             572#   4900*   5095*   5247*   5258*
UIPAR7= 177656             573#   3068    3248    3262    3477    4660*   4800*   5096*   5248*   5317*   5381*
UIPDR0= 177600             555#   3128    3351    3353    3366    3367    3380    3553    3700    3701*   3703    3708    3711
                          3791    4661*   4791*   5113    5164    5224    5235    5243*   5256*   5275*   5286    5298*   5374
                          5658
UIPDR1= 177602             556#   4662*   4792*
UIPDR2= 177604             557#   4663*   4793*
UIPDR3= 177606             558#   4664*   4794*   4890*
UIPDR4= 177610             559#   4670*   4682*   4891*   5508*   5672*
UIPDR5= 177612             560#   4801*   4892*
UIPDR6= 177614             561#   4893*   5257*
UIPDR7= 177616             562#   3131    3369    3383    3562    4665*   4795*   5244*   5252    5276*   5299*
USESTK= 000700             606#   2672    2850    5400    5523    5675    5785    5799
USEVEC  015574            2520#   4803
WBIT  = 000100            4998#   5005    5011    5026    5031    5038
WCNSSW= 000226             616#   6307
WLWHAM= 000305             617#
WWHAM = 000222             615#   4727
$APTHD  000234             654     660#
$ASTAT= ****** U          6068    6083
$ATYC   041444            6039    6041#
```

```
$ATY1   041420          6037#
$ATY3   041426          5983    6038#
$ATY4   041436          6040#   6484
$AUTOB  001134           701#
$BASE   001302           779#
$BDADR  001122           696#
$BDDAT  001126           698#
$BELL   001216           730#   6471    6510
$CDW1   001306           781#
$CDW2   001310           782#
$CHARC  041414          6000*   6010*   6017    6026*   6031#
$CKSWR= ****** U        6273
$CLR.T  040752          5880    5883#
$CMTAG  001100           684#   2578    2579    2587    2593    2594    2595
$CM1    = 000006         716#    717#    718#    719#    720#    721#    722#
$CM2    = 000014         716#    717#    718#    719#    720#    721#    722#
$CM3    = 000006         714#    716
$CM4    = 000005         722#    723#    724#    725#    726#    727#    728#
$CPUOP  001254           753#
$CRLF   001223           732#   5875    5999    6034    6479    6510    6538    6567    6574
$DBLK   042330          6192    6226    6234#
$DB22   042664          6347#   6609
$DDW0   001312           783#
$DDW1   001314           784#
$DDW10  001336           793#
$DDW11  001340           794#
$DDW12  001342           795#
$DDW13  001344           796#
$DDW14  001346           797#
$DDW15  001350           798#
$DDW2   001316           785#
$DDW3   001320           786#
$DDW4   001322           787#
$DDW5   001324           788#
$DDW6   001326           789#
$DDW7   001330           790#
$DDW8   001332           791#
$DDW9   001334           792#
$DEVCT  001236           744#
$DEVM   001304           780#
$DOAGN  040772          5857    5878    5885    5891#
$DTBL   042320          6195    6230#
$ENDAD  040762           673    2638    5887#   6498
$ENDCT  040610          2593    5859#
$ENULL  041036          5907#
$ENV    001246           749#   5978    6046    6070    6481
$ENVM   001247           750#   2630    5980    5985    6048
$EOP    040554          5814    5849#
$EOPCT  040602          2593*   5856#   5860
$ERFLG  001103           687#   6381    6413    6415    6421*   6443    6466*   6510    6737    6767    6801
$ERMAX  001115           693#   2596*   6415    6438*   6443
$ERROR  043300          2254    2587    6457#
$ERRPC  001116           694#   2189    2191    2193    2195    2197    2199    2202    2206    2208    2210    2212    2214
                        2216    2218    2220    2223    2225    2228    2230    2232    2234    2236    2239    2242    2244
                        2246    2248    2250    2252    2256    2258    2260    2262    2264    2265    2268    2270    2272
                        6473*   6474*   6475    6510    6535
```

```
SERRTB  001500            886#  6558  6562
SERTTL  001112            691#  5872  5876*  6472*  6510
SESCAP  001214            729#  2595*  6437*  6494  6496  6510
SETABL  001246            748#
SETEND  001352            666  801#
SFATAL  001230            741#  6074*
SFFLG   041664           6037*  6040*  6068  6077*  6085#
SFILLC  001156            712#  6003  6034
SFILLS  001155            711#  6034
SGDADR  001120            695#
SGDDAT  001124            697#
SGET42  040734           5877#
SGTSWR= ****** U          6272
SHD     = 000000           398
SHIBTS  000234            661#
SICNT   001104            688#  6428*  6429  6431*  6442
SILLUP  042612           6282  6298  6325#
SINTAG  001135            702#
SITEMB  001114            692#  6475*  6483  6510  6533
SLF     001224            733#  6034  6510
SLFLG   041663           6078*  6084#
SLOOP   041030           5899  5903#
SLPADR  001106            689#  2447  2611*  2645*  2818*  4583*  4983*  5352*  6419*  6435*  6440  6442
SLPERR  001110            690#  2612*  2644*  2701*  2721*  2757*  2767*  2778*  2786*  2792*  2800*  2820*  2831*
                         2885*  2894*  2900*  2905*  2918*  2964*  2971*  2977*  2984*  3027*  3032*  3038*  3064*
                         3070*  3095*  3101*  3127*  3133*  3168*  3183*  3203*  3229*  3244*  3264*  3289*  3304*
                         3324*  3350*  3365*  3385*  3421*  3438*  3466*  3483*  3508*  3525*  3551*  3568*  3595*
                         3604*  3613*  3630*  3639*  3648*  3663*  3672*  3681*  3698*  3707*  3716*  3737*  3745*
                         3753*  3784*  3805*  3820*  3835*  3850*  3865*  3880*  3895*  3910*  3925*  3940*  3962*
                         3976*  3992*  4025*  4063*  4071*  4110*  4127*  4162*  4183*  4218*  4243*  4257*  4281*
                         4294*  4395*  4402*  4406*  4411*  4558*  4589*  4622*  4666*  4766*  4805*  4833*  4901*
                         4951*  4984*  5002*  5008*  5014*  5020*  5034*  5041*  5097*  5100*  5199*  5222*  5321*
                         5353*  5382*  5393*  5407*  5416*  5425*  5433*  5444*  5455*  5465*  5476*  5526*  5538*
                         5550*  5564*  5578*  5592*  5607*  5621*  5661*  5685*  5694*  5703*  5711*  5721*  5732*
                         5742*  5753*  5791*  5801*  5815*  6419  6436*  6442  6493
SMADR1  001260            766#
SMADR2  001264            770#
SMADR3  001270            773#
SMADR4  001274            776#
SMAIL   001226            662  666  739#  2629  5978  6434  6481
SMAMS1  001256            760#
SMAMS2  001262            768#
SMAMS3  001266            771#
SMAMS4  001272            774#
SMBADR  000236            662#
SMFLG   041662           6038*  6044  6079*  6083#
SMSGAD  001242            746#  6054*  6057
SMSGLG  001244            747#  6059*
SMSGTY  001226            740#  6052  6060*  6072  6076*
SMTYP1  001257            761#
SMTYP2  001263            769#
SMTYP3  001267            772#
SMTYP4  001273            775#
SMXCNT  043276           6432  6442#
SNULL   001154            710#  6005  6034
SNWTST= 000001           2686#  2688  2734#  2736  2804#  2806  2834#  2836  2868#  2870  2922#  2924  2946#
```

```
                    2948    3010#   3012    3046#   3048    3077#   3079    3109#   3111    3149#   3151    3210#   3212
                    3270#   3272    3331#   3333    3400#   3402    3445#   3447    3489#   3491    3532#   3534    3583#
                    3585    3618#   3620    3651#   3653    3686#   3688    3720#   3722    3765#   3767    3994#   3996
                    4093#   4095    4164#   4166    4221#   4223    4298#   4300    4358#   4360    4565#   4567    4623#
                    4625    4771#   4773    4853#   4855    4967#   4969    5059#   5061    5204#   5206    5334#   5336
                    5491#   5493    5638#   5640    5767#   5769    5803#   5805
$OCNT   042110      6123*   6152*   6165#
$OCTVL  042766      6349    6374#
$OMODE  042112      6118*   6122*   6127    6130*   6141*   6167#
$OVER   043262      6394    6412    6420    6430    6439#
$PASS   001234       743#   2629*   2648    5853*   5854*   5865    5906    6426    6443
$PASTM  000242       664#
$PWRAD  042574      6321#
$PWRDN  042420      2591    6282#   6316
$PWRMG  042570      6319#
$PWRUP  042472      6292    6298#
$QUES   001222       731#   6034    6510
$RDCHR= ****** U    6273
$RDDEC= ****** U    6273
$RDLIN= ****** U    6273
$RDOCT= ****** U    6273
$REGAD  001160       714#
$REG0   001162       716#   2197    2206    2208    2210    2212    2214    2216    2218    2220    2225    2252    2256
                    2262    2268    2270    2272    6460*
$REG1   001164       717#   2189    2191    2206    2210    2214    2218    2220    2228    2230    2236    2244    2250
                    2254    2256    2260    2270    2272    6461*
$REG2   001166       718#   2191    2193    2197    2206    2214    2218    2220    2225    2234    2236    2246    2254
                    2265    2272    6462*
$REG3   001170       719#   2193    2225    2230    2234    2236    2250    2252    2265    6463*
$REG4   001172       720#   2272    6464*
$REG5   001174       721#   6465*
$RESRE  041100      5943#   6274
$RTNAD  041032      5905#
$RTRN   041026      2599    2601*   2606*   5881    5900#
$R2A  = ****** U    6275
$SAVRE  041042      5927#   6273
$SAVR6  042616      6291*   6299    6300*   6301*   6327#
$SCOPE  043004      2585    6390#
$SETUP= 000037       621#   2584    2585    2587    2589    2591    2593    2594    2595    2597    2611    2638    2641
                    5851    6323    6391    6458    6491    6498
$STUP = 177777       621#
$SVLAD  043226      6402    6433#
$SVPC = 000250       671#    676
$SWR  = 177400       389#    398     402     403     404     405     406     407     408     409     728     729     730
                    2594    2595    2597    2611    2612    2704    2756    2816    2847    2884    2935    2962    3026
                    3062    3093    3125    3167    3228    3288    3349    3419    3464    3506    3549    3595    3E30
                    3663    3698    3737    3778    4009    4108    4179    4235    4312    4374    4581    4652    4788
                    4869    4981    5077    5222    5350    5507    5655    5779    5814    5845    5852    5879    5892
                    5906    6322    6382    6383    6384    6385    6386    6393    6405    6407    6408    6413    6414
                    6415    6422    6423    6424    6436    6439    6442    6449    6450    6451    6452    6453    6469
                    6476    6488    6491    6510
$SWREG  001250       751#   2632
$SWRMK= 000200       389#    409     410     6386    6387    6409    6410
$TBIT   041034      2610*   5896*   5906#   6323*
$TESTN  001232       742#   6434*
$TIMES  001212       728#   2594*   3419*   3464*   3506*   3549*   4108*   4179*   4869*   5852*   6422*   6429    6432*
```

```
                    6442
STKB    001146       707#
STKS    001144       706#
STMP0   001176       722#  2232  2239  2248  3189* 3196* 3250* 3257* 3310* 3317* 3371* 3378* 3806*
                     3819  3821* 3834  3836* 3849  3851* 3864  3866* 3879  3881* 3894  3896* 3909
                     3911* 3924  3926* 3939  3941* 3954  4027* 4048  4253* 4268* 4328* 4433* 4443
                     4485  4497  4523  4609* 4614  4912* 4924  5004* 5005  5010* 5011  5016* 5017
                     5021* 5022  5023* 5026  5031  5037* 5038  5107* 5133* 5160* 5183* 5186  5229*
                     5255* 5282* 5305* 5308
STMP1   001200       723#  2220  2223  2232  3808* 3823* 3838* 3853* 3868* 3883* 3898* 3913* 3928*
                     3943* 3964* 3978* 4010* 4031  4033* 4040* 4254* 4269* 4329* 4415* 4418  4494
                     4689* 4698* 4704* 4732* 4740* 4745* 4752* 4755* 4756
STMP2   001202       724#  2220  2223  2232  2239  3809* 3824* 3839* 3854* 3869* 3884* 3899* 3914*
                     3929* 3944* 3965* 3979* 4037* 4255* 4270* 4330* 4376* 4456  4469* 4480* 4500
                     4513* 4519* 4610* 4611  4690* 4694* 4696  5446* 5447  5467* 5468  5595  5596*
                     5610* 5612  5723* 5724  5744* 5745
STMP3   001204       725#  2225  4045* 4461* 4471  4472* 4482  4505* 4515  4608* 4613* 4616* 4617
                     4709* 4728
STMP4   001206       726#  2212  2225  4046* 4339* 4353* 4462* 4473* 4506* 4715* 4718*
STMP5   001210       727#  2195  2199  2202  2212  3042* 3074* 3105* 3137* 3206* 3267* 3327* 3388*
                     3441* 3486* 3528* 3571* 4338* 4352* 4464* 4475* 4508* 4714* 4719*
STN     = 000054     389#   398  2670  2686  2704# 2723  2734  2753  2756# 2804  2813  2816# 2834
                     2844  2847# 2854  2868  2881  2884# 2922  2932  2935# 2946  2959  2962# 2989
                     3010  3023  3026# 3041  3046  3059  3062# 3073  3077  3090  3093# 3104  3109
                     3122  3125# 3136  3149  3164  3167# 3205  3210  3225  3228# 3266  3270  3285
                     3288# 3326  3331  3346  3349# 3387  3400  3416  3419# 3440  3445  3461  3464#
                     3485  3489  3503  3506# 3527  3532  3546  3549# 3570  3583  3592  3595# 3618
                     3627  3630# 3651  3660  3663# 3686  3695  3698# 3720  3734  2737# 3765  3775
                     3778# 3783  3994  4006  4009# 4072  4093  4105  4108# 4109  4164  4176  4179#
                     4221  4232  4235# 4298  4309  4312# 4358  4371  4374# 4565  4678  4581# 4623
                     4649  4652# 4771  4785  4788# 4834  4853  4866  4869# 4967  4978  4981# 4984
                     5043  5059  5074  5077# 5204  5219  5222# 5334  5347  5350# 5353  5477  5491
                     5504  5507# 5623  5638  5652  5655# 5754  5767  5776  5779# 5803  5814#
STPB    001152       709#  6023# 6034
STPFLG  001157       713#  5972  6034
STPS    001150       708#  6021  6034
STRAP   042340      2589   6243#
STRAP2  042362      6254#  6265
STRP    = 000012    6258#  6267#  6268#  6269#  6270#  6271#  6273  6274#  6275#  6276#  6277#
STRPAD  042374      6248   6265#
STSTM   000240       663#
STSTNM  001102       686#  2702*  2703  2886*  2887  3028*  3029  3785*  3786  4111*  4112  4985*  4986
                    5354*  5355  5851*  6381  6411  6433*  6434  6439  6443  6458  6468  6510
STYPBN= ****** U    6271
STYPDS  042114      6180#  6270
STYPE   041136      5972#  6065  6258  6266
STYPEC  041350      6002   6009  6016  6021#  6022
STYPEX  041416      6027   6029  6032#
STYPOC  041712      6121#  6267
STYPON  041726      6120   6123#  6269
STYPOS  041666      6116#  6268
SUNIT   001240       745#
SUNITM  000244       665#
SUSWR   001252       752#
SVECT1  001276       777#
SVECT2  001300       778#
```

```
$XTSTR  043024           6396#
$$GET4= 000001           5879#   5884
$OFILL  042111           6117*   6121*   6131    6166#
$40CAT= ****** U         6393    6478
.     = 044500            623#    627#    650     651#    653#    655#    671     672#    674#    676#    683#    734     849#
                         2362#   2536#   2582    2611    2612    2644    2645    4468    4479    4512    4723    5864#   5906
                         5908#   6034    6086#   6234#   6294    6326    6374#   6442    6443    6510    6626#
.$ASTA= ****** U         6038    6041
.$X   = 000234            650#    655
```

```
ADD18    2364#  3805   3820   3835   3850   3865   3880   3895   3910   3925   3940
AND      2364#  6734   6760   6764   6790   6794   6798
BYTADR   2364#  3595   3630   3663   3698
BYTEMG   3616#  3620   3653   3688
CMPPDR   2364#  5139   5163   5261   5285
COMMEN    540#
COUNT1   2364#  3420   3465   3507   3550
DUAL     2364#  3167   3228   3288   3349
DUALMA   2364#  5100   5222
DUALMG   3209#  3212   3272   3333
ENDCOM    540#
ENTRY    2364#  2700   2884   3026   3783   4109   4984   5353
ERROR     434#  2398   2401   2439   2441   2443   2486   2490   2517   2523   2725   2765   2776   2785   2790
          2799  2830   2855   2893   2899   2904   2915   2941   2970   2976   2983   2990   3043   3075   3106
          3138  3175   3207   3236   3268   3296   3328   3357   3389   3442   3487   3529   3572   3603   3611
          3638  3646   3671   3679   3706   3714   3751   3818   3833   3848   3863   3878   3893   3908   3923
          3938  3953   3975   3989   4038   4047   4069   4148   4152   4204   4208   4256   4271   4293   4331
          4340  4354   4442   4455   4470   4481   4496   4499   4514   4520   4535   4540   4545   4619   4716
          4724  4758   4810   4832   4929   4941   5007   5013   5019   5028   5033   5040   5147   5170   5189
          5269  5292   5311   5403   5405   5414   5423   5431   5440   5452   5462   5474   5485   5521   5536
          5548  5561   5575   5589   5604   5620   5631   5681   5683   5692   5701   5709   5718   5729   5739
          5750  5762   5788   5790   5798   5825   5829   6739   6741   6769   6771   6803   6805
ESCAPE    540#
GETPRI    540#  5892
GETSWR    540#  2641#
LOADPD   2364#  3796   5108   5112   5230   5234   5369   5373   5657
MGDUAL   5059#  5061   5206
MMLOAD   2364#  4113   4871   4987   5356
MSG1     2686#  2688
MSG10    2946#  2948
MSG11    3010#  3012
MSG12    3046#  3048
MSG13    3077#  3079
MSG14    3109#  3111
MSG15    3149#  3151
MSG16    3210#  3212
MSG17    3270#  3272
MSG20    3331#  3333
MSG21    3400#  3402
MSG22    3445#  3447
MSG23    3489#  3491
MSG24    3532#  3534
MSG25    3583#  3585
MSG26    3618#  3620
MSG27    3651#  3653
MSG30    3686#  3688
MSG31    3720#  3722
MSG32    3765#  3767
MSG33    3994#  3996
MSG345   4093#  4095   4166
MSG36    4221#  4223
MSG37    4298#  4300
MSG4     2734#  2736
MSG40    4358#  4360
MSG41    4565#  4567
MSG42    4623#  4625
```

```
MSG43    4771#  4773
MSG44    4853#  4855
MSG45    4967#  4969
MSG46    5059#  5061
MSG47    5204#  5206
MSG5     2804#  2806
MSG50    5334#  5336
MSG51    5491#  5493
MSG52    5638#  5640
MSG53    5767#  5769
MSG54    5803#  5805
MSG6     2834#  2836
MSG7     2868#  2870
MSG7A    2922#  2924
MULT      540#
NEWTST    540#  2686  2734  2804  2834  2868  2922  2946  3010  3046  3077  3109  3149  3210  3270
         3331   3400  3445  3489  3532  3583  3618  3651  3686  3720  3765  3994  4093  4164  4221
         4298   4358  4565  4623  4771  4853  4967  5059  5204  5334  5491  5638  5767  5803
PAGEF    2364#  4124  4180
PARMSG   3444#  3447
PDRMSG   3531#  3534
POP       540#  5948  6080  6081  6221  6309  6310
PUSH      540#  5928  6041  6043  6064  6180  6284  6290
READLG   2364#  4461  4472  4505
REPORT    540#
RESPMG   3045#  3048  3079  3111
RESPON   2364#  3030  3062  3093  3125
SAVTST   2364#  6458
SCOPE     435#  2753  2813  2844  2881  2932  2959  3023  3059  3090  3122  3164  3225  3285  3346
         3416   3461  3503  3546  3592  3627  3660  3695  3734  3775  4006  4105  4176  4232  4309
         4371   4578  4649  4785  4866  4978  5074  5219  5347  5504  5652  5776  5813  5850
SETACF   2364#  4312
SETPRI    540#
SETTRA   6258#  6267  6268  6269  6270  6273  6274  6275  6276
SETUP     540#  2577
SKIP      540#  2670  2723  2854  2989  3041  3073  3104  3136  3205  3266  3326  3387  3440  3485
         3527   3570  4072  4834  5043  5477  5623  5754
SLASH     540#
SPACE     540#
SSCOPE   2364#  6391
STARS     540#   647   649   656   669   679   734   737  2686  2699  2734  2751  2804  2811  2834
         2842   2868  2879  2922  2930  2946  2957  3010  3021  3046  3057  3077  3088  3109  3120
         3149   3162  3210  3223  3270  3283  3331  3344  3400  3414  3445  3459  3489  3501  3532
         3544   3583  3590  3618  3625  3651  3658  3686  3693  3720  3732  3765  3773  3994  4004
         4093   4103  4164  4174  4221  4230  4298  4307  4358  4369  4565  4576  4623  4647  4771
         4783   4853  4864  4967  4976  5059  5072  5204  5217  5334  5345  5491  5502  5638  5650
         5767   5774  5803  5812  5841  5912  5957  6036  6093  6170  6237  6280  6296  6338  6378
         6445
SWRSU     540#  2613#
TRACK    2364#  2752  2812  2843  2880  2931  2958  3022  3058  3089  3121  3163  3224  3284  3345
         3415   3460  3502  3545  3591  3626  3659  3694  3733  3774  4005  4104  4175  4231  4308
         4370   4577  4648  4784  4865  4977  5073  5218  5346  5503  5651  5775
TRACK1   2364#  2753  2813  2844  2881  2932  2959  3023  3059  3090  3122  3164  3225  3285  3346
         3416   3461  3503  3546  3592  3627  3660  3695  3734  3775  4006  4105  4176  4232  4309
         4371   4578  4649  4785  4866  4978  5074  5219  5347  5504  5652  5776
TRMTRP   6258#
```

```
TYPBIN    540#
TYPDEC    540#   5865    5872
TYPNAM    540#   2634
TYPNUM    540#
TYPOCS    540#
TYPOCT    540#
TYPTXT    540#   5861    5868
UPCODE   2364#   6303
USER      620#    803
ZEROER   2364#   6502
$SCMRE    677#    716     717     718     719     720     721
$SCMTM    677#    722     723     724     725     726     727
$SESCA    540#
$SNEWT    540#   2686    2734    2804    2834    2868    2922    2946    3010    3046    3077    3109    3149    3210    3270
         3331    3400    3445    3489    3532    3583    3618    3651    3686    3720    3765    3994    4093    4164    4221
         4298    4358    4565    4623    4771    4853    4967    5059    5204    5334    5491    5638    5767    5803
$SSET     6258#   6267    6268    6269    6270    6273    6274    6275    6276
$SSETM    2629#
$SSETU    2597#
$SSKIP    540#   2670    2723    2854    2989    3041    3073    3104    3136    3205    3266    3326    3387    3440    3485
         3527    3570    4072    4834    5043    5477    5623    5754
.EQUAT    389#    430
.HEADE    389#
.KT11     389#    540
.SETUP    389#    621
.SWRHI    389#    398
.SWRLO    410#
.$ACT1    389#    667
.$APTB    389#    735#
.$APTH    389#    645
.$APTY    389#   6034
.$CATC    389#    621
.$CMTA    389#    677
.$D820    389#   6336
.$EOP     389#   5839
.$ERRO    389#   6443
.$POWE    389#   6278
.$SAVE    389#   5910
.$SCOP    389#   6376
.$TRAP    389#   6235
.$TYPD    389#   6168
.$TYPE    389#   5955
.$TYPO    389#   6091

. ABS.   044500     000

 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:DQKTAA,DSKZ:DQKTAA.SEQ/CRF/SOL=DSKZ:DQKTAA.P11
 RUN-TIME: 24 21 2 SECONDS
 RUN-TIME RATIO: 628/48=12.9
 CORE USED:  32K  (64 PAGES)
```

F12