11/70

UNIBUS MAP DIAGNOSTIC

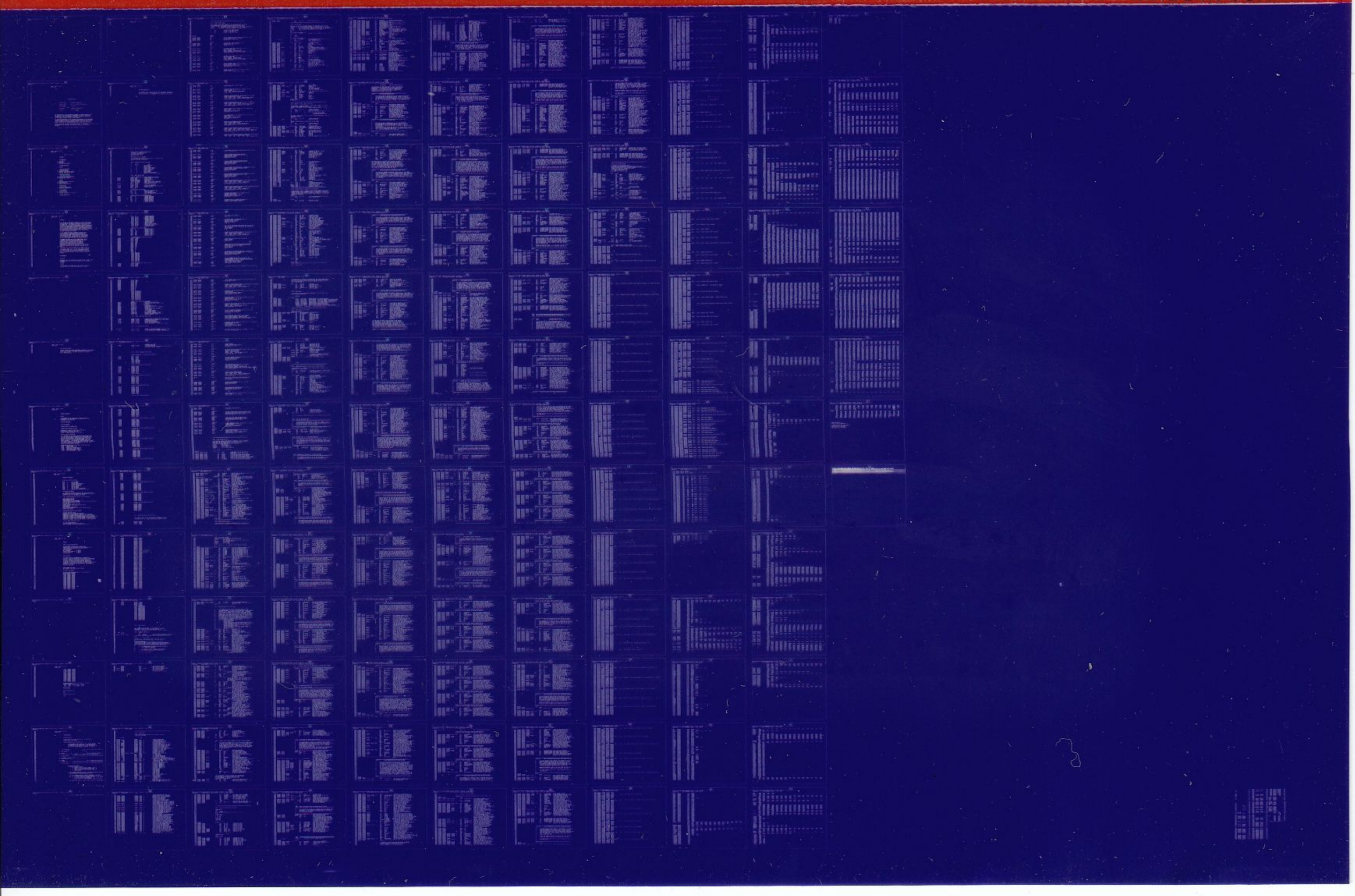
MD-11-DEKBF-B

EP DEKBF B DL A

FICHE 1 OF 1

NOV 1976

MADE IN USA



```
                        .REM %
```

```
                        .MAIN.  MACY11 27(657)                  PAGE 1
                        DEKBFA.MAN

                                .REM @
```

IDENTIFICATION

PRODUCT CODE:	MAINDEC-11-DEKBF-B-D
PRODUCT NAME:	PDP-11/70 UNIBUS MAP DIAGNOSTIC
DATE CREATED:	21-JUL-75
MAINTAINER:	DIAGNOSTIC ENGINEERING
AUTHORS:	DALE A. ROEDGER

TABLE OF CONTENTS

1. ABSTRACT

THIS PROGRAM IS DESIGNED TO BE RUN ON A PDP11/70 ON WHICH THE CPU, CACHE, AND MEMORY MANAGEMENT DIAGNOSTIC PROGRAMS HAVE BEEN RUN. THE PROGRAM WILL DETECT ALL ERRORS THAT ORIGINATE WITH THE MAP BOX AND PROVIDE LOOPING CAPABILITIES SO THAT THE FIELD SERVICE ENGINEER CAN VERIFY THE FAILURES. THERE MAY BE SOME CASES, SUCH AS THE CACHE REGISTER DATA PATH, AND CACHE MEMORY DATA PATH, WHERE INTERACTION BETWEEN MODULES PROHIBITS CLOSE ISOLATION, BUT THE FAILING FUNCTION WILL BE CALLED OUT SO THE FIELD SERVICE ENGINEER CAN COMPLETE THE ISOLATION PROCESS.

IF THE PROGRAM CATCHES AN ERROR IN AN EARLY TEST AND IS ALLOWED TO CONTINUE RUNNING THROUGH THE LATER TESTS THE ERROR INDICATIONS FROM THOSE LATER TESTS MAY BE INVALID. THIS IS DUE TO THE STRUCTURE OF THE PROGRAM, WHICH ASSUMES THAT ALL AREAS TESTED PRIOR TO THE CURRENT TEST ARE FUNCTIONING PROPERLY.

THE ERROR TYPE OUTS WILL BE IN TABLE FORMAT, WITH A MESSAGE INDICATING THE CLASS OF ERROR, A HEADER IDENTIFYING EACH COLUMN AND A REPORT OF ALL PERTINENT DATA. WHEN THE TEST CAN PRODUCE MORE THAN ONE ERROR CONDITION, A SUMMARY OF ERRORS WILL BE GIVEN AT THE END OF THAT TEST CONSISTING OF: THE LOGICAL 'AND' AND 'OR' OF THE DATA PREVIOUSLY REPORTED AND THE NUMBER OF ERRORS IN THIS TEST.
(SEE SECTION 6.3 FOR AN EXAMPLE OF THE ERROR TYPEOUTS.)

THE PROGRAM LOADS "044" INTO THE MICRO BREAK REGISTER (1777770) SO THAT A SYNC PULSE IS GENERATED ON PIN # AE1 SLOT10 EVERY TIME A "NOP" IS EXECUTED. THIS SHOULD HELP IN ISOLATING THE EXACT TIMING OF BAD OR MISSING SIGNALS.

2. REQUIREMENTS

2.1 EQUIPMENT

THE BASIC PDP-11/70 COMPUTER, INCLUDING THE CPU, CACHE, MEMORY MANAGEMENT, AND AN LA-30 OR EQUIVALENT DEVICE FOR ERROR MESSAGES.

2.2 STORAGE

THIS PROGRAM WILL REQUIRE 8K TO LOAD BUT WILL UTILIZE ALL EXISTING CORE FOR A DUAL ADDRESSING TEST OF MEMORY

156 FROM THE UNIBUS.

2.3 PRELIMINARY PROGRAMS

THE CPU, CACHE, AND MEMORY MANAGEMENT DIAGNOSTICS SHOULD BE RUN BEFORE THIS PROGRAM. THE MEMORY DIAGNOSTIC SHOULD AT LEAST, MAKE A QUICK VERIFY OF THE AREA OF MEMORY THIS PROGRAM WILL LOAD AND RUN IN.

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.

4. STARTING PROCEDURE

4.1 STARTING ADDRESS

PROGRAM STARTS AT ADDRESS 200

4.2 PROGRAM AND/OR OPERATOR ACTION

PROGRAM WILL IDENTIFY ITSELF AND AT THE END OF EACH PASS WILL INDICATE THE TOTAL NUMBER OF ERRORS OCCURRING ON THAT PASS.

4.3 SPECIAL STARTING PROCEDURE

IF IT APPEARS THAT THE CACHE IS CAUSING SOME TROUBLE AND YOU STILL WANT TO RUN THIS PROGRAM, IT IS POSSIBLE TO RUN WITH THE CACHE DISABLED. SIMPLY LOAD THE CACHE CONTROL REGISTER (17777746) WITH THE DESIRED NUMBER. THEN LOAD THE PC (17777707) WITH THE STARTING ADDRESS (200) AND PRESS "CONTINUE". THE PROGRAM WILL NOW RUN NORMALLY EXCEPT THAT CERTAIN TESTS WILL BE SKIPPED SINCE THE CACHE IS DISABLED. THIS FACT IS INDICATED IN THE ABSTRACT OF EACH TEST THAT CHECKS THE CACHE CONTROL REGISTER.

DEFINITION OF THE BITS IN THE CACHE CONTROL REGISTER:
BIT00 -DISABLE TRAPS
BIT01 -DISABLE UNIBUS TRAPS
BIT02 -FORCE MISS ON READ, GROUP 0
BIT03 -FORCE MISS ON READ, GROUP 1
BIT04 -FORCE REPLACEMENT IN GROUP 0
BIT05 -FORCE REPLACEMENT IN GROUP 1

5. OPERATING PROCEDURE

5.1 OPERATIONAL SWITCH SETTINGS

SW15	1=	HALT ON ERROR
SW14	1=	LOOP ON TEST
SW13	1=	INHIBIT ERROR TYPEOUTS
SW12	1=	INHIBIT TRACE TRAP
SW11	1=	INHIBIT ITERATIONS
SW10	1=	BELL ON ERROR
SW09	1=	LOOP ON ERROR
SW08	1=	LOOP ON TEST IN SWR<06:00>
SW07	1=	INHIBIT MULTIPLE ERROR TYPE OUTS

5.2 SUB-ROUTINE ABSTRACTS

ALL SUBROUTINE ABSTRACTS APPEAR IN THE CODE BEFORE THEIR EXPANSION AND IN THE DOCUMENT THAT IMMEDIATELY FOLLOWS THIS. BELOW IS A LIST OF THE SUBROUTINE TITLES.

5.2.1 MACRO LIBRARY SUBROUTINES (FOUND IN MOST PROGRAMS)

SCOPE HANDLER ROUTINE
ERROR HANDLER ROUTINE
ERROR MESSAGE TYPE OUT ROUTINE
CONVERT 16-BIT VIRTUAL ADDRESSES TO 22-BIT PHYSICAL ADDRESSES
SAVE AND RESTORE R0-R5 ROUTINES
TYPE ROUTINE
BINARY TO OCTAL (ASCII) AND TYPE
CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
TRAP DECODER
POWER DOWN AND UP ROUTINES
DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE
END OF PASS ROUTINE

5.2.2 SUBROUTINES UNIQUE TO THIS PROGRAM

TURN OFF AND SAVE T-BIT
RESTORE T-BIT TO ITS PREVIOUS CONDITION
SUBROUTINE TO LOG AND REPORT TIMEOUTS OF MAP REGISTERS
SUBROUTINE TO REPORT MAP REGISTERS THAT WILL NOT HOLD ZERO
SUBROUTINE TO REPORT DUAL ADDRESSING WHEN LOADING A MAP REGISTER
SUBROUTINE TO REPORT COUNT PATTERN ERRORS IN MAP REGISTERS
SUBROUTINE TO REPORT COUNT PATTERN ERRORS ON UNIBUS DATA PATH
SUBROUTINE TO REPORT COUNT PATTERN ERRORS IN CACHE REGISTERS

5.2.3 TRAP AND ABORT HANDLER ROUTINES

CPU TRAP HANDLER ROUTINE
CACHE TRAPS AND ABORTS HANDLER ROUTINE
MEMORY MANAGEMENT TRAPS AND ABORTS HANDLER ROUTINE

6. ERRORS

6.1 ERROR HALTS AND DESCRIPTION

WHEN AN ERROR IS DETECTED AN 'ERROR' (EMT)
INSTRUCTION IS EXECUTED AND THE 'ERROR HANDLER ROUTINE'
CHECKS THE SWITCH REGISTER FOR MODE SELECTED.
THE PROGRAM WILL:

HALT ON ERROR	IF SW15=1
INHIBIT ERROR TYPE OUT	IF SW13=1
RING BELL ON ERROR	IF SW10=1
LOOP ON ERROR	IF SW9=1

6.2 ERROR RECOVERY

IF SWITCH 9 IS UP, THE PROGRAM WILL LOOP BACK TO
THE POINT WHERE THE INSTRUCTION THAT CAUSED THE ERROR WAS
EXECUTED, WITHOUT ALLOWING ANY OF THE CONDITIONS TO CHANGE.
THIS WILL PROVIDE THE TIGHTEST POSSIBLE SCOPE LOOP.
IF SWITCH 9 IS DOWN, EACH ERROR WILL BE REPORTED AND
LOGGED AND, AT THE END OF EACH TEST, A SUMMARY OF ALL ERRORS
OCCURRING IN THAT TEST WILL BE PROVIDED. THE SUMMARY
CONSISTS OF THE LOGICAL AND AND OR OF THE ADDRESS AND/OR
DATA THAT WAS WRONG.

6.3 SAMPLE ERROR TYPE OUTS

SEE "$ERRTB:" FOR SAMPLE ERROR TYPEOUTS.

6.3.1 MULTIPLE TYPE ERRORS

THE FOLLOWING REGISTERS TIMED OUT WHEN REFERENCED

REG.ADR	TESTNO	ERRORPC
170210	000001	015226
170212	000001	015232
170214	000001	015232
170216	000001	015232
170230	000001	015232
170232	000001	015232
170234	000001	015232
170236	000001	015232
170250	000001	015232
170252	000001	015232
170254	000001	015232
170256	000001	015232
170270	000001	015232
170272	000001	015232
170274	000001	015232
170276	000001	015232
170310	000001	015232

```
   325                                          170312  000001  015232
```

```
                    170314  000001  015232
                    170316  000001  015232
                    170330  000001  015232
                    170332  000001  015232
                    170334  000001  015232
                    170336  000001  015232
                    170350  000001  015232
                    170352  000001  015232
                    170354  000001  015232
                    170356  000001  015232
                    170370  000001  015232
                    170372  000001  015232
                    170374  000001  015232
                    170376  000001  015232

                    SUMMARY OF MAP REGISTERS THAT TIMED OUT ON READ
                    REGADRS REGADRS
                     "OR"    "AND"   #ERRORS TESTNO  ERRORPC
                    170376  170210      32  000001  010530
```

7. RESTRICTIONS

7.1 STARTING RESTRICTIONS

NONE

7.2 OPERATING RESTRICTIONS

NONE

8. MISCELLANEOUS

8.1 EXECUTION TIME

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

8.2 ADDRESS GENERATION IN THE PDP-11/70

THE FOLLOWING IS AN EXAMPLE OF HOW A MEMORY ADDRESS IS GENERATED BY THE UNIBUS MAP. THIS ASSUMES THAT THE ADDRESS ORIGINATES IN THE CPU BUT THE PROCESS CAN APPLY TO ANY UNIBUS ADDRESS, STARTING AT LINE C2.

```
A.  VIRTUAL ADDRESS                          15 14 13 12 11 10 09 08 07 06 05 04 03

A1. PAGE NUMBER (0-7)                        15 14 13

A2. OFFSET                                                12 11 10 09 08 07 06 05 04 03
B.  P.A.R.[PAGE NO.] +  15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00
                        ------------------------------------------------
C.  PHYSICAL ADDR.      21 20 19 18 17 16 15 14 13 12 11 10 09 08 07 06 05 04 03

C1. 17XXXXXX=> U.B.ADR. 21 20 19 18
C2. MAPPING REG.NO.(0-36)              17 16 15 14 13

C3. OFFSET                                                12 11 10 09 08 07 06 05 04 03
D.  MAP REG.[NO.]    +  21 20 19 18 17 16 15 14 13 12 11 10 09 08 07 06 05 04 03
                        ------------------------------------------------
E.  PHYSICAL ADDR.      21 20 19 18 17 16 15 14 13 12 11 10 09 08 07 06 05 04 03
```

DESCRIPTION OF LINES:

A: VIRTUAL ADDRESS (16 BITS)
- A1: UPPER 3 BITS OF VIRTUAL ADDRESS, USED TO SELECT A PAGE ADDRESS REGISTER (PAR)
- A2: LOWER 13 BITS OF VIRTUAL ADDRESS, ADDED TO SELECTED PAR

B: PAGE ADDRESS REGISTER (16 BITS), IN ADDITION PROCESS THIS GET LEFT SHIFTED 6 BITS BEFORE ADDITION TO A2

C: PHYSICAL ADDRESS CREATED BY MEMORY MANAGEMENT, (22 BITS)
- C1: IF UPPER 4 BITS ARE ALL ONES THEN BITS <17:00> GO OU
- C2: IF MAP RELOCATION IS ENABLED THEN BITS <17:13> SELEC OF THE 36 (OCTAL) MAP REGISTERS.
- C3: LOWER 13 BITS OF UNIBUS ADDRESS, ADDED TO SELECTED M

D: MAP REGISTER (22 BITS), ADDED TO BITS <12:00> OF UNIBUS ADDRE

```
   416                                  E: PHYSICAL ADDRESS GENERATED BY UNIBUS MAP AND SENT TO THE CACH
```

```
                                .MAIN.  MACY11 27(657)                  PAGE 10
                                DEKBFA.MAN
```

9. PROGRAM DESCRIPTION

THE DOCUMENT THAT FOLLOWS THIS ONE HAS A PARAGRAPH DESCIRBING EACH OF THE TESTS. THE PARAGRAPH WILL INDICATE IF THE TEST IS RUN CONDITIONALLY ON THE STATUS ON THE CACHE CONTROL REGISTER.

@
.END.

```
                        ;
                        .TITLE  PDP-11/70 UNIBUS MAP DIAGNOSTIC
                        ;*COPYRIGHT (C) MARCH 15, 1975
                        ;*DIGITAL EQUIPMENT CORP.
                        ;*MAYNARD, MASS. 01754
                        ;*
                        ;*PROGRAM BY DALE A. ROEDGER
                        ;*
                        ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
                        ;*PACKAGE (MAINDEC-11-DZQAC-A3).
                        ;*

                        .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

                        .SBTTL  BASIC DEFINITIONS

                        ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
001100                  STACK=  1100            ;;FIRST ADDRESS OF THE STACK
001100                  KERSTK= STACK           ;;KERNEL STACK
000700                  SUPSTK= STACK-200       ;;SUPERVISOR STACK
000600                  USESTK= STACK-300       ;;USER STACK
                        .EQUIV  EMT,ERROR       ;;BASIC DEFINITION OF ERROR CALL
                        .EQUIV  IOT,SCOPE       ;;BASIC DEFINITION OF SCOPE CALL
177776                  PS=     177776          ;;PROCESSOR STATUS WORD
                        .EQUIV  PS,PSW
177774                  STKLMT= 177774          ;;STACK LIMIT REGISTER
177772                  PIRQ=   177772          ;;PROGRAM INTERRUPT REQUEST REGISTER
177570                  SWR=    177570          ;;SWITCH REGISTER
177570                  DISPLAY=SWR

                        ;*MISCELLANEOUS DEFINITIONS
000011                  HT=     11              ;;CODE FOR HORIZONTAL TAB
000012                  LF=     12              ;;CODE LINE FEED
000015                  CR=     15              ;;CODE CARRIAGE RETURN
000200                  CRLF=   200             ;;CODE FOR CARRIAGE RETURN-LINE FEED

                        ;*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
```

```
                                R7=     %7              ;;GENERAL REGISTER
                                .EQUIV  R0,R10          ;;GENERAL REGISTER
                                .EQUIV  R1,R11          ;;GENERAL REGISTER
                                .EQUIV  R2,R12          ;;GENERAL REGISTER
                                .EQUIV  R3,R13          ;;GENERAL REGISTER
                                .EQUIV  R4,R14          ;;GENERAL REGISTER
                                .EQUIV  R5,R15          ;;GENERAL REGISTER
                                .EQUIV  R6,SP           ;;STACK POINTER
                                .EQUIV  SP,KSP          ;;KERNEL STACK POINTER
                                .EQUIV  SP,SSP          ;;SUPERVISOR STACK POINTER
                                .EQUIV  SP,USP          ;;USER STACK POINTER
                                .EQUIV  R7,PC           ;;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
```

F02


```
                        BIT11=  4000
                        BIT10=  2000
                        BIT09=  1000
                        BIT08=  400
                        BIT07=  200
                        BIT06=  100
                        BIT05=  40
                        BIT04=  20
                        BIT03=  10
                        BIT02=  4
                        BIT01=  2
                        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
000114                  CACHVEC=114             ;;CACHE ERROR INTERRUPT VECTOR
000240                  PIRQVEC=240             ;;PROGRAM INTERRUPT REQUEST VECTOR
000250                  MMVEC=  250             ;;MEMORY MANAGEMENT VECTOR

                        .SBTTL  CACHE   REGISTER DEFINITIONS

177740                  LOADRS = 177740         ;;LOWER 16 BITS OF ADDRESS THAT CAUSED ERROR
177742                  HIADRS = 177742         ;;UPPER SIX BITS OF ADDRESS THAT CAUSED ERROR
177744                  MEMERR = 177744         ;;CACHE ERROR REGISTER
177746                  CONTRL = 177746         ;;MEMORY CONTROL REGISTER
177750                  MAINT  = 177750         ;;MEMORY MAINTENENCE REGISTER
177752                  HITMIS = 177752         ;;HIT MISS REGISTER "1" IMPLIES HIT IN CACHE

                        .SBTTL  CPU REGISTER DEFINITIONS

177760                  SIZELO = 177760         ;;MEMORY SIZE REGISTER NUMBER TO PUT INTO A PAR
                                                ;;TO GET TO THE LAST 32 WORDS OF MEMORY
177762                  SIZEHI = 177762         ;;HIGH SIZE REGISTER, RESERVED FOR FUTURE USE
```

```
                                                ;;CURRENTLY ALL ZERO
            177764              SYSTID = 177764 ;;SYSTEM ID REGISTER
            177766              CPUERR = 177766 ;;CPU ERROR REGISTER HOLDS CONDITION THAT CAUSED
                                                ;;THE TRAP TO ERRVEC (000004)



                                .SBTTL  MEMORY MANAGEMENT DEFINITIONS


                                ;*MEMORY MANAGEMENT STATUS REGISTER ADDRESSES

            177572              MMR0=   177572
            177574              MMR1=   177574
            177576              MMR2=   177576
            172516              MMR3=   172516
                                .EQUIV  MMR0,SR0
                                .EQUIV  MMR1,SR1
                                .EQUIV  MMR2,SR2
                                .EQUIV  MMR3,SR3

                                ;*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 "D" PAGE DESCRIPTOR REGISTORS

            177620              UDPDR0= 177620
            177622              UDPDR1= 177622
            177624              UDPDR2= 177624
            177626              UDPDR3= 177626
            177630              UDPDR4= 177630
            177632              UDPDR5= 177632
            177634              UDPDR6= 177634
            177636              UDPDR7= 177636

                                ;*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

                                ;*USER "D" PAGE ADDRESS REGISTERS
```

```
                        
                        
177660                  UDPAR0= 177660
177662                  UDPAR1= 177662
177664                  UDPAR2= 177664
177666                  UDPAR3= 177666
177670                  UDPAR4= 177670
177672                  UDPAR5= 177672
177674                  UDPAR6= 177674
177676                  UDPAR7= 177676
                        
                        ;*SUPERVISOR "I" PAGE DESCRIPTOR REGISTERS
                        
172200                  SIPDR0= 172200
172202                  SIPDR1= 172202
172204                  SIPDR2= 172204
172206                  SIPDR3= 172206
172210                  SIPDR4= 172210
172212                  SIPDR5= 172212
172214                  SIPDR6= 172214
172216                  SIPDR7= 172216
                        
                        ;*SUPERVISOR "D" PAGE DESCRIPTOR REGISTERS
                        
172220                  SDPDR0= 172220
172222                  SDPDR1= 172222
172224                  SDPDR2= 172224
172226                  SDPDR3= 172226
172230                  SDPDR4= 172230
172232                  SDPDR5= 172232
172234                  SDPDR6= 172234
172236                  SDPDR7= 172236
                        
                        ;*SUPERVISOR "I" PAGE ADDRESS REGISTERS
                        
172240                  SIPAR0= 172240
172242                  SIPAR1= 172242
172244                  SIPAR2= 172244
172246                  SIPAR3= 172246
172250                  SIPAR4= 172250
172252                  SIPAR5= 172252
172254                  SIPAR6= 172254
172256                  SIPAR7= 172256
                        
                        ;*SUPERVISOR "D" PAGE ADDRESS REGISTERS
                        
172260                  SDPAR0= 172260
172262                  SDPAR1= 172262
172264                  SDPAR2= 172264
172266                  SDPAR3= 172266
172270                  SDPAR4= 172270
172272                  SDPAR5= 172272
172274                  SDPAR6= 172274
172276                  SDPAR7= 172276
                        
                        ;*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 "D" PAGE DESCRIPTOR REGISTERS

                        172320          KDPDR0= 172320
                        172322          KDPDR1= 172322
                        172324          KDPDR2= 172324
                        172326          KDPDR3= 172326
                        172330          KDPDR4= 172330
                        172332          KDPDR5= 172332
                        172334          KDPDR6= 172334
                        172336          KDPDR7= 172336

                                        ;*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

                                        ;*KERNEL "D" PAGE ADDRESS REGISTERS

                        172360          KDPAR0= 172360
                        172362          KDPAR1= 172362
                        172364          KDPAR2= 172364
                        172366          KDPAR3= 172366
                        172370          KDPAR4= 172370
                        172372          KDPAR5= 172372
                        172374          KDPAR6= 172374
                        172376          KDPAR7= 172376



                                        .SBTTL  UNIBUS MAP REGISTER DEFINITIONS

                                        ;*THE LOWER 16 BITS OF THE MAP REGISTERS ARE LABELED 'MAPLXX'
                                        ;*THE UPPER 6 BITS OF THE MAP REGISTERS ARE LABELED 'MAPHXX'

                        170200          MAPL00 = 170200
                        170202          MAPH00 = 170202
                        170204          MAPL01 = 170204
                        170206          MAPH01 = 170206
```

```
  770          170210                  MAPLO2 = 170210
  771          170212                  MAPHO2 = 170212
  772          170214                  MAPLO3 = 170214
  773          170216                  MAPHO3 = 170216
  774          170220                  MAPLO4 = 170220
  775          170222                  MAPHO4 = 170222
  776          170224                  MAPLO5 = 170224
  777          170226                  MAPHO5 = 170226
  778          170230                  MAPLO6 = 170230
  779          170232                  MAPHO6 = 170232
  780          170234                  MAPLO7 = 170234
  781          170236                  MAPHO7 = 170236
  782          170240                  MAPL10 = 170240
  783          170242                  MAPH10 = 170242
  784          170244                  MAPL11 = 170244
  785          170246                  MAPH11 = 170246
  786          170250                  MAPL12 = 170250
  787          170252                  MAPH12 = 170252
  788          170254                  MAPL13 = 170254
  789          170256                  MAPH13 = 170256
  790          170260                  MAPL14 = 170260
  791          170262                  MAPH14 = 170262
  792          170264                  MAPL15 = 170264
  793          170266                  MAPH15 = 170266
  794          170270                  MAPL16 = 170270
  795          170272                  MAPH16 = 170272
  796          170274                  MAPL17 = 170274
  797          170276                  MAPH17 = 170276
  798          170300                  MAPL20 = 170300
  799          170302                  MAPH20 = 170302
  800          170304                  MAPL21 = 170304
  801          170306                  MAPH21 = 170306
  802          170310                  MAPL22 = 170310
  803          170312                  MAPH22 = 170312
  804          170314                  MAPL23 = 170314
  805          170316                  MAPH23 = 170316
  806          170320                  MAPL24 = 170320
  807          170320                  MAPH24 = 170320
  808          170324                  MAPL25 = 170324
  809          170326                  MAPH25 = 170326
  810          170330                  MAPL26 = 170330
  811          170332                  MAPH26 = 170332
  812          170334                  MAPL27 = 170334
  813          170336                  MAPH27 = 170336
  814          170340                  MAPL30 = 170340
  815          170342                  MAPH30 = 170342
  816          170344                  MAPL31 = 170344
  817          170346                  MAPH31 = 170346
  818          170350                  MAPL32 = 170350
  819          170352                  MAPH32 = 170352
  820          170354                  MAPL33 = 170354
  821          170356                  MAPH33 = 170356
  822          170360                  MAPL34 = 170360
  823          170362                  MAPH34 = 170362
  824          170364                  MAPL35 = 170364
  825          170366                  MAPH35 = 170366
```

```
                        170370                  MAPL36 = 170370
                        170372                  MAPH36 = 170372
                        170374                  MAPL37 = 170374
                        170376                  MAPH37 = 170376
                                                .EQUIV  MAPL00,MAPL0
                                                .EQUIV  MAPH00,MAPH0
                                                .EQUIV  MAPL01,MAPL1
                                                .EQUIV  MAPH01,MAPH1
                                                .EQUIV  MAPL02,MAPL2
                                                .EQUIV  MAPH02,MAPH2
                                                .EQUIV  MAPL03,MAPL3
                                                .EQUIV  MAPH03,MAPH3
                                                .EQUIV  MAPL04,MAPL4
                                                .EQUIV  MAPH04,MAPH4
                                                .EQUIV  MAPL05,MAPL5
                                                .EQUIV  MAPH05,MAPH5
                                                .EQUIV  MAPL06,MAPL6
                                                .EQUIV  MAPH06,MAPH6
                                                .EQUIV  MAPL07,MAPL7
                                                .EQUIV  MAPH07,MAPH7


                                                .SBTTL  TRAP CATCHER

                        000000                          .=0
                                                ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
                                                ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
                                                ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS

                                                .SBTTL  STARTING ADDRESS(ES)
                        000200                          .=200

000200  000137  010000                          JMP     @#START         ;;JUMP TO STARTING ADDRESS OF PROGRAM
                                                ;;*****************************************************************


                                                .SBTTL          ACT11 HOOKS

                                                ;*THE FOLLOWING LOCATIONS ARE SETUP TO BE USED WITH ACT11
                                                ;*
                                                ;*LOCATION 46 WILL CONTAIN THE ADDRESS OF THE LOCICAL
                                                ;*END OF THE PROGRAM.
                                                ;*LOCATION 52 IS USED TO SPECIFY PROGRAM OPERATING REQUIREMENTS
                                                ;*AND/OR RESTRICTIONS. THIS IS ACCOMPLISHED BY SETTING VARIOUS BITS
                                                ;*TO A ONE OR A ZERO. THE BITS USED AND THERE MEANING ARE:
                                                ;*
                                                ;*      BIT 15=1 PROGRAM SHOULD BE POWER FAILED WHILE RUNNING
                                                ;*           =0 NO POWER FAIL DESIRED
                                                ;*
                                                ;*      BIT 14=1 PROGRAM RUN TIME IS MEMORY SIZE DEPENDENT
                                                ;*           =0 RUN TIME IS NOT MEMORY SIZE DEPENDENT
                                                ;*
                                                ;*      BITS 13-0  MUST BE ZERO'S
```

```
   882           000204                  $SVPC=.                        ;;SAVE LOCATION COUNTER
   883           000046                  .=46                           ;;SET LOCATION COUNTER
   884  000046   023100                  .WORD   $ENDAD                 ;;SET LOC.46 TO ADDRESS $ENDAD
   885           000052                  .=52                           ;;SET LOCATION COUNTER
   886  000052   000000                  .WORD   0                      ;;SET LOC.52 TO ZERO
   887           000204                  .=$SVPC                        ;; RESTORE LOCATION COUNTER
```

```
                                ;;*****************************************************************
                                .SBTTL  COMMON TAGS

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

                001100                  .=1100

001100                          $CMTAG:                 ;;START OF COMMON TAGS
001100  000000                  $PASS:  .WORD   0       ;;CONTAINS PASS COUNT
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
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  OF 'GOOD' DATA
001122  000000                  $BDADR: .WORD   0       ;;CONTAINS  OF 'BAD' DATA
001124  000000                  $GDDAT: .WORD   0       ;;CONTAINS 'GOOD' DATA
001126  000000                  $BDDAT: .WORD   0       ;;CONTAINS 'BAD' DATA
001130  000000  000000  000000          .WORD   0,0,0   ;;RESERVED--NOT TO BE USED
001136  177560                  $TKS:   177560          ;;TTY KBD STATUS
001140  177562                  $TKB:   177562          ;;TTY KBD BUFFER
001142  177564                  $TPS:   177564          ;;TTY PRINTER STATUS REG.
001144  177566                  $TPB:   177566          ;;TTY PRINTER BUFFER REG.
001146  000                     $NULL:  .BYTE   0       ;;CONTAINS NULL CHARACTER FOR FILLS
001147  002                     $FILLS: .BYTE   2       ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
001150  012                     $FILLC: .BYTE   12      ;;INSERT FILL CHARS. AFTER A "LINE FEED"
001151  000                     $TPFLG: .BYTE   0       ;;"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
001152  000000                  $REGAD: .WORD   0       ;;CONTAINS THE  FROM
                                                        ;;WHICH  ($REGO) WAS OBTAINED
001154  000000                  $REGO:  .WORD   0       ;;CONTAINS (($REGAD)+0)
001156  000000                  $REG1:  .WORD   0       ;;CONTAINS (($REGAD)+2)
001160  000000                  $REG2:  .WORD   0       ;;CONTAINS (($REGAD)+4)
001162  000000                  $REG3:  .WORD   0       ;;CONTAINS (($REGAD)+6)
001164  000000                  $REG4:  .WORD   0       ;;CONTAINS (($REGAD)+10)
001166  000000                  $REG5:  .WORD   0       ;;CONTAINS (($REGAD)+12)
001170  000000                  $TMP0:  .WORD   0       ;;USER DEFINED
001172  000000                  $TMP1:  .WORD   0       ;;USER DEFINED
001174  000000                  $TMP2:  .WORD   0       ;;USER DEFINED
001176  000000                  $TMP3:  .WORD   0       ;;USER DEFINED
001200  000000                  $TMP4:  .WORD   0       ;;USER DEFINED
001202  000000                  $TMP5:  .WORD   0       ;;USER DEFINED
001204  000000                  $TIMES: 0               ;;MAX. NUMBER OF ITERATIONS
001206  000000                  $ESCAPE:0               ;;ESCAPE ON ERROR
001210  177607  000377           $BELL:  .ASCIZ  <207><377><377> ;;CODE FOR BELL
001214  077                     $QUES:  .ASCII  /?/     ;;QUESTION MARK
001215  015                     $CRLF:  .ASCII  <15>    ;;CARRIAGE RETURN
001216  000012                  $LF:    .ASCIZ  <12>    ;;LINE FEED
001220  000000                  PADRSL: .WORD   0       ;HOLDS THE LOWER 16 BITS OF A 22 BIT
                                                        ;ADDRESS GENERATED FOR TYPE OUT.
001222  000000                  PADRSH: .WORD   0       ;HOLDS THE UPPER 6 BITS OF A 22 BIT
```

```
944
945  001224  000000                ADRAND: .WORD   0               ;ADDRESS GENERATED FOR TYPE OUT
946  001226  000000                ADDROR: .WORD   0               ;LOGICAL AND OF FAILING ADDRESSES
947  001230  000000                DATAND: .WORD   0               ;LOGICAL OR OF FAILING ADDRESSES
948  001232  000000                DATAOR: .WORD   0               ;LOGICAL AND OF BAD DATA
949  001234  000000                PATAND: .WORD   0               ;LOGICAL OR OF BAD DATA
950  001236  000000                PATTOR: .WORD   0               ;LOGICAL AND OF PATTERN LOADED
951  001240  000000                LOWEST: .WORD   0               ;LOGICAL OR OF PATTERN LOADED
952                                                               ;HOLDS NUMBER TO PUT IN PAR TO CAUSE THE
953  001242  000000                HIGEST: .WORD   0               ;LOWEST USEABLE MAP REGISTER TO RESPOND
954                                                               ;HOLDS NUMBER TO PUT IN PAR TO CAUSE THE
955  001244  000000                LREGL:  .WORD   0               ;HIGHEST USEABLE MAP REGISTER TO RESPOND
956                                                               ;HOLDS I/O PAGE ADDR OF LOW 16 BITS OF
957  001246  000000                LREGU:  .WORD   0               ;THE LOWEST USEABLE MAP REGISTER
958                                                               ;HOLDS I/O PAGE ADDR OF HIGH 16 BITS OF
959  001250  000000                HREGL:  .WORD   0               ;OF THE LOWEST USEABLE MAP REGISTER
960                                                               ;HOLDS I/O PAGE ADDR OF LOW 16 BITS OF
961  001252  000000                HREGU:  .WORD   0               ;THE HIGHEST USEABLE MAP REGISTER
962                                                               ;HOLDS I/O PAGE ADDR OF HIGH 16 BITS OF
963  001254  000000                ERRCNT: .WORD   0               ;THE HIGHEST USEABLE MAP REGISTER
964  001256  000000                CNTR:   .WORD   0               ;MULTIPLE ERROR ERROR COUNTER
965  001260  000000                FLAG:   .WORD   0               ;AUXILIARY COUNTER
966  001262  000000                TESTNO: .WORD   0               ;FLAG TO INDICATE TO LAST PROGRAM PASS N
967  001264  000000                CPUEXP: .WORD   0               ;HOLDS TEST NUMBER FOR ERROR TYPE OUTS
968  001266  000000                PCPUER: .WORD   0               ;HOLDS THE EXPECTED CPU ERROR CODE
969  001270  000000                PLOADR: .WORD   0               ;HOLDS RECEIVED CPU ERROR CONDITION
970  001272  000000                PHIADR: .WORD   0               ;HOLDS LOWER 16 BITS OF CACHE ADDR
971  001274  000000                PPARER: .WORD   0               ;HOLDS UPPER 6 BITS OF CACHE ADDR
972  001276  000000                PCONTR: .WORD   0               ;HOLDS RECEIVED PARITY ERROR CONDITION
973  001300  000000                PMAINT: .WORD   0               ;HOLDS CONTENTS OF CONTROL REGISTER
974  001302  000000                BADPC:  .WORD   0               ;HOLDS CONTENTS OF MAINTENENCE REGISTER
975  001304  000000                OLDPC:  .WORD   0               ;HOLDS PC OF INST THAT CAUSED TRAP
976  001306  000000                OLDPS:  .WORD   0               ;HOLDS THE RETURN ADDRESS AFTER A TRAP
977  001310  000000                OLDPSW: .WORD   0               ;HOLDS THE OLD PROCESSOR STATUS
978  001312  000000                PMMR0:  .WORD   0               ;HOLDS OLD PSW FOR TBITRESTORE
979  001314  000000                PMMR1:  .WORD   0               ;HOLDS CONTENTS OF MMR0 AFTER TRAP
980  001316  000000                PMMR2:  .WORD   0               ;HOLDS CONTENTS OF MMR1 AFTER TRAP
981  001320  000000                RSIZE:  .WORD   0               ;HOLDS CONTENTS OF MMR2 AFTER TRAP
982  001322  000000                RETRY:  .WORD   0               ;WILL HOLD P.A.R. DATA FOR TOP OF MEMORY
983  001324  000000                NXTTST: .WORD   0               ;RETRY FLAG IN CASE OF PARITY ABORTS
984                                                               ;LOCATION TO HOLD ESCAPE ADDRESS ON
985  001326  000200                DATA:   .WORD   200             ;PARITY ERRORS.
                                                                  ;PATTERN TO BE USED TO LOAD INTO MEMORY
```

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

                                .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


001330                          $ERRTB:
                                ;ITEM1
001330  023162                  EM1                     ;NOT THE CORRECT CPU TRAP CONDITION THRU ERRVEC (#004)
001332  031425                  DH1                     ;RECEIVD EXPECTD TESTNO  PC AT ABORT
001334  034360                  DT1                     ;PCPUER,CPUEXP,TESTNO,BADPC,0
001336  035134                  DF1                     ; 0, 0, 0, 0

                                ;ITEM 2
001340  023244                  EM2                     ;UNEXPECTED CPU TRAP THRU ERRVEC (#004)
001342  031471                  DH2                     ;RECEIVD TESTNO  PC AT ABORT
001344  034372                  DT2                     ;PCPUER,TESTNO,BADPC
001346  035140                  DF2                     ; 0, 0, 0

                                ;ITEM 3
001350  023313                  EM3                     ;UNEXPECTED CACHE PARITY ERROR THRU CACHVEC (#114)
                                                        ;WILL RETRY TEST ONCE
001352  031525                  DH3                     ;PARITY  ADDRESS   MAINTEN CONTROL
                                                        ;CONDITN REFERENCD REGISTR REGISTR TESTNO PC AT ABORT
001354  034402                  DT3                     ;PPARER,LOADDR,PMAINT,PCONTR,TESTNO,BADPC,0
001356  035143                  DF3                     ; 0, 2, 0, 0, 0, 0

                                ;ITEM 4
001360  023414                  EM4                     ;UNEXPECTED MAIN MEMORY PARITY ERROR THRU CACHVEC (#114)
                                                        ;WILL RETRY TEST ONCE
001362  031525                  DH3                     ;PARITY  ADDRESS   MAINTEN CONTROL
                                                        ;CONDITN REFERENCD REGISTR REGISTR TESTNO PC AT ABORT
001364  034402                  DT3                     ;PPARER,LOADDR,PMAINT,PCONTR,TESTNO,BADPC,0
001366  035143                  DF3                     ; 0, 2, 0, 0, 0, 0

                                ;ITEM 5
001370  023523                  EM5                     ;MEMORY MANAGEMENT TRAP, MEMORY MANAGEMENT STATUS REGISTERS
001372  031655                  DH5                     ;STATUS  AUTOI/D VIRTADR
                                                        ;REGISTR REGISTR REGISTR TESTNO  PC AT ABORT
001374  034432                  DT5                     ;PMMR0,PMMR1,PMMR2,TESTNO,BADPC,0
001376  035151                  DF5                     ; 0, 0, 0, 0, 0

                                ;ITEM 6
001400  023631                  EM6                     ;SUMMARY OF MAP REGISTERS THAT TIMED OUT ON READ
001402  031761                  DH6                     ;REGADRS REGADRS
                                                        ; "OR"    "AND"  #ERRORS TESTNO  ERRORPC
```

```
1042  001404  034446                  DT6                     ;ADDROR,ADRAND,ERRCNT,TESTNO,$ERRPC,0
1043  001406  035156                  DF6                     ; 0, 0, 1, 0, 0
1044
1045                                  ;ITEM 7
1046  001410  023711                  EM7                     ;SUMMARY OF CACHE REGISTERS THAT TIMED OUT ON READ
1047  001412  031761                  DH6                     ;REGADRS REGADRS
1048                                                          ; "OR"    "AND"  #ERRORS TESTNO  ERRORPC
1049  001414  034446                  DT6                     ;ADDROR,ADRAND,ERRCNT,TESTNO,$ERRPC,0
1050  001416  035156                  DF6                     ; 0, 0, 1, 0, 0
1051
1052                                  ;ITEM 10
1053  001420  023773                  EM10                    ;SUMMARY OF MAP REGISTERS NOT HOLDING ZERO IN LOW 16 BITS
1054  001422  032051                  DH10                    ;REGADRS REGADRS RECEIVD RECEIVD
1055                                                          ; "OR"    "AND"   "OR"    "AND"   #ERRORS TESTNO  ERRORPC
1056  001424  034462                  DT10                    ;ADDROR,ADRAND,DATAOR,DATAND,ERRCNT,TESTNO,$ERRPC,0
1057  001426  035163                  DF10                    ; 0, 0, 0, 0, 1, 0, 0
1058
1059                                  ;ITEM 11
1060  001430  024064                  EM11                    ;SUMMARY OF MAP REGISTERS NOT HOLDING ZERO IN UPPER 6 BITS
1061  001432  032051                  DH10                    ;REGADRS REGADRS RECEIVD RECEIVD
1062                                                          ; "OR"    "AND"   "OR"    "AND"   #ERRORS TESTNO  ERRORPC
1063  001434  034462                  DT10                    ;ADDROR,ADRAND,DATAOR,DATAND,ERRCNT,TESTNO,$ERRPC,0
1064  001436  035163                  DF10                    ; 0, 0, 0, 0, 1, 0, 0
1065
1066                                  ;ITEM 12
1067  001440  024156                  EM12                    ;POSSIBLE ERROR IN MAP REGISTER DATA PATH (MAP REG 00)
1068  001442  032201                  DH12                    ;COUNT    COUNT
1069                                                          ;EXPECTD RECEIVD TESTNO  ERRORPC
1070  001444  034502                  DT12                    ;$REG2,$REG0,TESTNO,$ERRPC,0
1071  001446  035172                  DF12                    ; 0, 0, 0, 0
1072
1073                                  ;ITEM 13
1074  001450  024244                  EM13                    ;NOW PROBABLE ERROR IN MAP REGISTER DATA PATH (MAP REG 20)
1075  001452  032201                  DH12                    ;COUNT    COUNT
1076                                                          ;EXPECTD RECEIVD TESTNO  ERRORPC
1077  001454  034514                  DT13                    ;$REG3,$REG1,TESTNO,$ERRPC,0
1078  001456  035172                  DF12                    ; 0, 0, 0, 0
1079
1080                                  ;ITEM 14
1081  001460  024336                  EM14                    ;SUMMARY OF DUAL ADDRESSING ERRORS ON LOADING MAP REGISTERS
1082  001462  032257                  DH14                    ;REGLOAD REGLOAD REGDUAL REGDUAL
1083                                                          ; "OR"    "AND"   "OR"    "AND"   #ERRORS TESTNO
1084  001464  034526                  DT14                    ;ADDROR,ADRAND,DATAOR,DATAND,ERRCNT,TESTNO,0
1085  001466  035176                  DF14                    ; 0, 0, 0, 0, 1, 0
1086
1087                                  ;ITEM 15
1088  001470  024431                  EM15                    ;SUMMARY OF COUNT PATTERN FAILURES IN LOWER 16 BITS OF MAP REGIS
1089  001472  032376                  DH15                    ;MAPREG  MAPREG  EXPECTD EXPECTD RECEIVD RECEIVD
1090                                                          ; "OR"    "AND"   "OR"    "AND"   "OR"    "AND"   #ERRORS TESTNO
1091  001474  034544                  DT15                    ;ADDROR,ADRAND,PATTOR,PATAND,DATAOR,DATAND,ERRCNT,TESTNO,0
1092  001476  035204                  DF15                    ; 0, 0, 0, 0, 0, 0, 1, 0
1093
1094                                  ;ITEM 16
1095  001500  024535                  EM16                    ;SUMMARY OF COUNT PATTERN FAILURES IN UPPER 6 BITS OF MAP REGIST
1096  001502  032376                  DH15                    ;MAPREG  MAPREG  EXPECTD EXPECTD RECEIVD RECEIVD
1097                                                          ; "OR"    "AND"   "OR"    "AND"   "OR"    "AND"   #ERRORS TESTNO
```

```
1098  001504  034544                DT15                ;ADDROR,ADRAND,PATTOR,PATAND,DATAOR,DATAND,ERRCNT,TESTNO,0
1099  001506  035204                DF15                ; 0, 0, 0, 0, 0, 0, 1, 0
1100
1101                                ;ITEM 17
1102  001510  024640                EM17                ;COULD NOT CLEAR CACHE CONTROL REGISTER
1103                                                    ;POSSIBLE ERROR IN CACHE REGISTER DATA PATH
1104  001512  032555                DH17                ;RECEIVD TESTNO  ERRORPC
1105  001514  034566                DT17                ;$REG0,TESTNO,$ERRPC,0
1106  001516  035214                DF17                ; 0, 0, 0
1107
1108                                ;ITEM 20
1109  001520  024762                EM20                ;COULD NOT CLEAR CACHE MAINTENENCE REGISTER
1110                                                    ;POSSIBLE ERROR IN CACHE REGISTER DATA PATH
1111  001522  032555                DH17                ;RECEIVD TESTNO  ERRORPC
1112  001524  034576                DT20                ;$REG1,TESTNO,$ERRPC,0
1113  001526  035214                DF17                ; 0, 0, 0
1114
1115                                ;ITEM 21
1116  001530  025110                EM21                ;COULD NOT READ 177740 FROM CACHE LO ADDRESS REG (LOADRS)
1117                                                    ;POSSIBLE ERROR IN CACHE REGISTER DATA PATH
1118  001532  032555                DH17                ;RECEIVD TESTNO  ERRORPC
1119  001534  034566                DT17                ;$REG0,TESTNO,$ERRPC,0
1120  001536  035214                DF17                ; 0, 0, 0
1121
1122                                ;ITEM 22
1123  001540  025254                EM22                ;COULD NOT READ 000003 FROM CACHE HI ADDRESS REG (HIADRS)
1124                                                    ;POSSIBLE ERROR IN CACHE REGISTER DATA PATH
1125  001542  032555                DH17                ;RECEIVD TESTNO  ERRORPC
1126  001544  034566                DT17                ;$REG0,TESTNO,$ERRPC,0
1127  001546  035214                DF17                ; 0, 0, 0
1128
1129                                ;ITEM 23
1130  001550  025420                EM23                ;SUMMARY OF COUNT PATTERN FAILURES IN CACHE CONTROL REGISTER
1131  001552  032605                DH23                ;EXPECTD EXPECTD RECEIVD RECEIVD
1132                                                    ; "OR"    "AND"   "OR"    "AND"   #ERRORS TESTNO
1133  001554  034606                DT23                ;PATTOR,PATAND,DATAOR,DATAND,ERRCNT,TESTNO,0
1134  001556  035217                DF23                ; 0, 0, 0, 0, 1, 0
1135
1136                                ;ITEM 24
1137  001560  025514                EM24                ;SUMMARY OF COUNT PATTERN FAILURES IN CACHE MAINTENENCE REGISTER
1138  001562  032605                DH23                ;EXPECTD EXPECTD RECEVID RECEIVD
1139                                                    ; "OR"    "AND"   "OR"    "AND"   #ERRORS TESTNO
1140  001564  034606                DT23                ;PATTOR,PATAND,DATAOR,DATAND,ERRCNT,TESTNO,0
1141  001566  035217                DF23                ; 0, 0, 0, 0, 1, 0
1142
1143                                ;ITEM 25
1144  001570  025614                EM25                ;REFERENCED MAP REGISTER 0 WITH ADDRESS ONE BIT
1145                                                    ;DIFFERENT THAN 770200
1146  001572  032724                DH25                ;ADDRUSED  BITDIFF TESTNO  ERRORPC
1147  001574  034624                DT25                ;$REG4,$REG0,TESTNO,$ERRPC,0
1148  001576  035225                DF25                ;3, 4, 0, 0
1149
1150                                ;ITEM 26
1151  001600  025721                EM26                ;REFERENCED CACHE LOW ADDRESS REGISTER WITH
1152                                                    ;ADDRESS ONE BIT DIFFERENT THAN 777740
1153  001602  032724                DH25                ;ADDRUSED  BITDIFF TESTNO  ERRORPC
```

```
1154  001604  034624                  DT25            ;$REG4,$REG0,TESTNO,$ERRPC,0
1155  001606  035225                  DF25            ;3, 4, 0, 0
1156
1157                                  ;ITEM 27
1158  001610  025721                  EM26            ;REFERENCED CACHE LO ADDRESS REGISTER WITH
1159                                                  ;ONE BIT DIFFERENT THAN 777740
1160  001612  032766                  DH27            ;ADDRUSED  TESTNO  ERRORPC
1161  001614  034636                  DT27            ;$REG1,TESTNO,$ERRPC
1162  001616  035231                  DF27            ;3, 0, 0
1163
1164                                  ;ITEM 30
1165  001620  026042                  EM30            ;CAN'T GET TO MAIN MEMORY FROM UNIBUS WITH THE MAP OFF
1166                                                  ;SO I'LL JUMP TO THE SIZE JUMPER TEST FOR VERIFICATION
1167  001622  033020                  DH30            ;TESTNO  ERRORPC
1168  001624  034646                  DT30            ;TESTNO,$ERRPC,0
1169  001626  035234                  DF30            ;0, 0
1170
1171                                  ;ITEM 31
1172  001630  026216                  EM31            ;SUMMARY OF COUNT PATTERN FAILURES ON THE UNIBUS DATA  PATH
1173  001632  032605                  DH23            ;EXPECTD EXPECTD RECEIVD RECEIVD
1174                                                  ; "OR"    "AND"   "OR"    "AND"   #ERRORS TESTNO
1175  001634  034606                  DT23            ;PATTOR,PATAND,DATAOR,DATAND,ERRCTN,TESTNO,0
1176  001636  035217                  DF23            ; 0, 0, 0, 0, 1, 0
1177
1178                                  ;ITEM 32
1179  001640  026310                  EM32            ;UNIBUS MAP IS RELOCATING WHEN NOT ENABLED
1180  001642  033020                  DH30            ;TESTNO  ERRORPC
1181  001644  034646                  DT30            ;TESTNO,$ERRPC,0
1182  001646  035234                  DF30            ; 0, 0
1183
1184                                  ;ITEM 33
1185  001650  026362                  EM33            ;CANNOT USE ANY OF THE MAP REGISTERS OR PHYSICAL
1186                                                  ;ADDRESS BIT14 IS STUCK LOW, MUST RESTART PROGRAM
1187                                                  ;IF YOU DON'T LOOP ON THIS PROBLEM.
1188  001652  033020                  DH30            ;TESTNO  ERRORPC
1189  001654  034646                  DT30            ;TESTNO,$ERRPC,0
1190  001656  035234                  DF30            ; 0, 0
1191
1192                                  ;ITEM 34
1193  001660  026537                  EM34            ;THE NUMBER OF MAP REGISTERS REMOVED BY JUMPER SETTING
1194                                                  ;DOES NOT AGREE WITH THE NUMBER FOUND TO BE MISSING.
1195  001662  033040                  DH34            ;REMOVED MISSING TESTNO  ERRORPC
1196  001664  034654                  DT34            ;ERRCNT,CNTR,TESTNO,$ERRPC,0
1197  001666  035236                  DF34            ; 0, 0, 0, 0
1198
1199                                  ;ITEM 35
1200  001670  026710                  EM35            ;THE SIZE JUMPERS ON THE UNIBUS MAP ARE NOT SET
1201                                                  ;IN THEIR DEFAULT POSITION.  THIS WOULD ALLOW UNIBUS
1202                                                  ;ADDRESSES 000000 TO 757776 TO REFERNECE MAIN MEMORY
1203                                                  ;THEIR CURRENT SETTING ALLOWS ONLY:
1204  001672  033100                  DH35            ;LOWEST  HIGHEST TESTNO  ERRORPC
1205  001674  034666                  DT35            ;LOWEST,HIGEST,TESTNO,$ERRPC,0
1206  001676  035242                  DF35            ; 4, 4, 0, 0
1207
1208                                  ;ITEM 36
1209  001700  027175                  EM36            ;MAP REGISTER UNDER TEST DID NOT RESPOND IN DUAL MAPPING TEST
```

```
1210  001702  033140                DH36                ;TESTNO  ERRORPC UNIBUS ADDRESS OF MAP REGISTER UNDER TEST
1211  001704  034700                DT36                ;TESTNO,$ERRPC,$REG0,0
1212  001706  035246                DF36                ; 0, 0, 3
1213
1214                                ;ITEM 37
1215  001710  027272                EM37                ;SUMMARY OF UNIBUS ADDRESS ERRORS, WITH MAP RELOCATION DISABLED
1216  001712  032605                DH23                ;EXPECTD EXPECTD RECEIVD RECEIVD
1217                                                    ; "OR"    "AND"   "OR"    "AND"  #ERRORS TESTNO
1218  001714  034710                DT37                ;ADDROR,ADRAND,DATAOR,DATAND,ERRCNT,TESTNO
1219  001716  035251                DF37                ; 0, 0, 0, 0, 1, 0
1220
1221                                ;ITEM 40
1222  001720  027375                EM40                ;MAIN MEMORY TIME OUT OVER THE UNIBUS DID NOT OCCUR PROPERLY.
1223  001722  033232                DH40                ;CONDITN CONDITN
1224                                                    ;EXPECTD RECEIVD TESTNO  ERRORPC
1225  001724  034726                DT40                ;CPUEXP,PCPUER,TESTNO,$ERRPC,0
1226  001726  035257                DF40                ; 0, 0, 0, 0
1227
1228                                ;ITEM 41
1229  001730  027470                EM41                ;RELOCATION THROUGH THE MAP WAS NOT CORRECT, FULL ADD.
1230  001732  033312                DH41                ;CORRECT ADDRESS
1231                                                    ;ADDRESS FETCHED TESTNO  ERRORPC
1232  001734  034740                DT41                ;$REG2,$REG1,TESTNO,$ERRPC
1233  001736  035263                DF41                ; 0, 0, 0, 0
1234
1235                                ;ITEM 42
1236  001740  027552                EM42                ;RELOCATION THRU THE MAP WAS NOT CORRECT, CARRY PROPAGATION
1237  001742  033372                DH42                ;CORRECT   EXPECTD RECEIVD
1238                                                    ;ADDRESS   DATA    FROM UB TESTNO  ERRORPC
1239  001744  034752                DT42                ;$REG1,$REG3,$REG2,TESTNO,$ERRPC
1240  001746  035267                DF42                ; 3, 0, 0, 0, 0
1241
1242                                ;ITEM 43
1243  001750  027645                EM43                ;THE TOP OF MEMORY IS DIFFERENT THAN THE SIZE JUMPERS
1244  001752  033472                DH43                ;SIZJUMP TOPFOUND TESTNO ERRORPC
1245  001754  034766                DT43                ;SIZELO,RSIZE,TESTNO,$ERRPC,0
1246  001756  035274                DF43                ; 0, 0, 0, 0
1247
1248                                ;ITEM 44
1249  001760  027732                EM44                ;PARITY REPORTING THRU THE MAP IS NOT CORRECT
1250  001762  033532                DH44                ;CONDITN CONDITN ADDRESS   MAINTEN CONTROL
1251                                                    ;EXPECTD RECEIVD REFERENCD REGISTR REGISTR TESTNO  ERRORPC
1252  001764  035000                DT44                ;$TMP4,PPARER,PLOADR,PMAINT,PCONTR,TESTNO,$ERRPC,0
1253  001766  035300                DF44                ; 0, 0, 2, 0, 0, 0, 0
1254
1255                                ;ITEM 45
1256  001770  030007                EM45                ;MAIN MEMORY TIME OUT OVER THE UNIBUS DID NOT OCCUR PROPERLY.
1257                                                    ;TEST BEING RUN OVER UNIBUS
1258  001772  033232                DH40                ;CONDITN CONDITN
1259                                                    ;EXPECTD RECEIVD TESTNO  ERRORPC
1260  001774  034726                DT40                ;CPUEXP,PCPUER,TESTNO,$ERRPC,0
1261  001776  035257                DF40                ; 0, 0, 0, 0
1262
1263                                ;ITEM 46
1264  002000  030142                EM46                ;RELOCATION THROUGH THE MAP WAS NOT CORRECT, FULL ADD.
1265                                                    ;TEST BEING RUN OVER UNIBUS
```

```
1266  002002  033312                DH41            ;CORRECT ADDRESS
1267                                                  ;ADDRESS FETCHED TESTNO  ERRORPC
1268  002004  034740                DT41            ;$REG2,$REG1,TESTNO,$ERRPC
1269  002006  035263                DF41            ; 0, 0, 0, 0
1270
1271                                ;ITEM 47
1272  002010  030264                EM47            ;RELOCATION THRU THE MAP WAS NOT CORRECT, CARRY PROPAGATION
1273                                                  ;TEST BEING RUN OVER UNIBUS
1274  002012  033372                DH42            ;CORRECT   EXPECTD RECEIVD
1275                                                  ;ADDRESS   DATA    FROM UB TESTNO  ERRORPC
1276  002014  034752                DT42            ;$REG1,$REG3,$REG2,TESTNO,$ERRPC
1277  002016  035267                DF42            ; 3, 0, 0, 0, 0
1278
1279                                ;ITEM 50
1280  002020  030417                EM50            ;THE TOP OF MEMORY IS DIFFERENT THAN THE SIZE JUMPERS
1281                                                  ;TEST BEING RUN OVER UNIBUS
1282  002022  033472                DH43            ;SIZJUMP TOPFOUND TESTNO
1283  002024  034766                DT43            ;SIZELO,RSIZE,TESTNO,0
1284  002026  035274                DF43            ; 0, 0, 0
1285
1286
1287                                ;ITEM 51
1288  002030  030544                EM51            ;PARITY REPORTING THRU THE MAP IS NOT CORRECT
1289                                                  ;TEST BEING RUN OVER UNIBUS
1290  002032  033532                DH44            ;CONDITN CONDITN ADDRESS   MAINTEN CONTROL
1291                                                  ;EXPECTD RECEIVD REFERENCD REGISTR REGISTR TESTNO  ERRORPC
1292  002034  035000                DT44            ;$TMP4,PPARER,PLOADR,PMAINT,PCONTR,TESTNO,$ERRPC,0
1293  002036  035300                DF44            ; 0, 0, 2, 0, 0, 0, 0
1294
1295                                ;ITEM 52
1296  002040  030661                EM52            ;SUMMARY OF DUAL MAPPING ERRORS
1297  002042  032605                DH23            ;EXPECTD EXPECTD RECEIVD RECEIVD
1298                                                  ; "OR"     "AND"   "OR"    "AND"   #ERRORS TESTNO
1299  002044  034710                DT37            ;ADDROR,ADRAND,DATAOR,DATAND,$ERRPC,TESTNO,0
1300  002046  035251                DF37            ; 0, 0, 0, 0, 1, 0
1301
1302  002050                        ER200:          ;THIS IS THE STARTING POINT FOR ERROR MESSAGES
1303                                                  ;201 THRU 377.  THEY ARE USED FOR MULTIPLE
1304                                                  ;ERROR MESSAGES.
1305
1306                                ;ITEM 201
1307  002050  030720                EM201           ;THE FOLLOWING REGISTERS TIMED OUT WHEN READ
1308  002052  033676                DH201           ;REGADRS TESTNO ERRORPC
1309  002054  035020                DT201           ;$REGO,TESTNO,$ERRPC,0
1310  002056  035307                DF201           ; 0, 0, 0
1311
1312                                ;ITEM 202
1313  002060  030774                EM202           ;THE FOLLOWING MAP REGISTERS WILL NOT CLEAR
1314  002062  033726                DH202           ;REGADRS DATAREC TESTNO  ERRORPC.
1315  002064  035030                DT202           ;$REGO,$TMP0,TESTNO,$ERRPC,0
1316  002066  035312                DF202           ; 0, 0, 0, 0
1317
1318                                ;ITEM 203
1319  002070  031047                EM203           ;THE FOLLOWING ARE DUAL ADDRESSING ERRORS IN THE UNIBUS MAP
1320  002072  033766                DH203           ;MAPREG  MAPREG
1321                                                  ;TESTING DUALED  TESTNO  ERRORPC
```

```
1322  002074  035042                DT203           ;$REG0,$REG1,$TESTNO,$ERRPC,0
1323  002076  035316                DF203           ; 0, 0, 0, 0
1324
1325                                 ;ITEM 204
1326  002100  031142                EM204           ;THE COUNT PATTERN THRU THE MAP REGISTERS FAILED
1327  002102  034045                DH204           ;REGADRS PATTERN EXPECTD RECEIVD TESTNO  ERRORPC
1328  002104  035054                DT204           ;$REG0,$REG2,$REG4,$REG3,TESTNO,$ERRPC,0
1329  002106  035322                DF204           ; 0, 0, 0, 0, 0, 0
1330
1331                                 ;ITEM 205
1332  002110  031222                EM205           ;UNIBUS DATA PATH COUNT PATTERN FAILURE
1333  002112  034125                DH205           ;EXPECTD RECEIVD ADDRSLOAD TESTNO  ERRORPC
1334  002114  035072                DT205           ;$REG1,$REG0,$REG2,TESTNO,$ERRPC,0
1335  002116  035330                DF205           ;0, 0, 3, 0, 0
1336
1337                                 ;ITEM 206
1338  002120  031271                EM206           ;UNIBUS ADDRESSING ERRORS, MAP RELOCATION DISABLED
1339  002122  034177                DH206           ;ADDRESS ADDRESS
1340                                                 ;EXPECTD RECEIVD TESTNO ERRORPC
1341  002124  035106                DT206           ;$REG0,$REG3,TESTNO,$ERRPC,0
1342  002126  035335                DF206           ; 0, 0, 0, 0
1343
1344                                 ;ITEM 207
1345  002130  031353                EM207           ;COUNT PATTERN FAILURES IN CACHE REGISTERS
1346  002132  034257                DH207           ;REGISTR EXPECTD RECEIVD
1347                                                 ;ADDRESS  DATA    DATA   TESTNO  ERRORPC
1348  002134  035120                DT207           ;$REG0,$REG2,$REG3,TESTNO,$ERRPC,0
1349  002136  035341                DF207           ; 0, 0, 0, 0, 0
1350
1351
1352
1353                                 ;;*****************************************************************
1354
1355                                 .SBTTL  SCOPE HANDLER ROUTINE
1356
1357                                 ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
1358                                 ;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
1359                                 ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
1360                                 ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
1361                                 ;*SW14=1         LOOP ON TEST
1362                                 ;*SW11=1         INHIBIT ITERATIONS
1363                                 ;*SW09=1         LOOP ON ERROR
1364                                 ;*SW08=1         LOOP ON TEST IN SWR<6:0>
1365                                 ;*CALL
1366                                 ;*      SCOPE           ;;SCOPE=IOT
1367
1368  002140                         $SCOPE:
1369  002140  005037  001322                CLR     RETRY           ;CLEAR RETRY FLAG AN THE START OF
1370                                                                 ;EACH TEST
1371  002144  005037  001254                CLR     ERRCNT          ;CLEAR THE MULTIPLE ERROR COUNTER
1372  002150  005037  001232                CLR     DATAOR          ;LOCATION FOR LOGICAL OR OF BAD DATA
1373  002154  005037  001226                CLR     ADDROR          ;LOCATION FOR LOGICAL OR OF ADDRESS
1374  002160  005037  001236                CLR     PATTOR          ;LOCATION FOR LOGICAL OR OF PATTERN LOADED
1375  002164  012700  177777                MOV     #-1,R0          ;LOAD -1 INTO R0 TO INITIALIZE LOGICAL AND LOCS
1376  002170  010037  001230                MOV     R0,DATAND       ;LOCATION FOR LOGICAL AND OF BAD DATA
1377  002174  010037  001224                MOV     R0,ADRAND       ;LOCATION FOR LOGICAL AND OF ADDRESS
```

```
1378  002200  010037  001234                    MOV     R0,PATAND       ;LOCATION FOR LOGICAL AND OF PATTERN LOADED
1379  002204  006137  177570                    ROL     @#SWR           ;;LOOP ON PRESENT TEST?
1380  002210  100514                            BMI     $OVER           ;;YES IF SW14=1
1381                                    ;#####START OF CODE FOR THE XOR TESTER#####
1382  002212  000416                  $XTSTR:   BR      6$              ;;IF RUNNING ON THE "XOR" TESTER CHANGE
1383                                                                    ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
1384  002214  013746  000004                    MOV     @#ERRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
1385  002220  012737  002240  000004            MOV     #5$,@#ERRVEC    ;;SET FOR TIMEOUT
1386  002226  005737  177060                    TST     @#177060        ;;TIME OUT ON XOR?
1387  002232  012637  000004                    MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
1388  002236  000466                            BR      $SVLAD          ;;GO TO THE NEXT TEST
1389  002240  022626                    5$:     CMP     (SP)+,(SP)+     ;;CLEAR THE STACK AFTER A TIME OUT
1390  002242  012637  000004                    MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
1391  002246  000426                            BR      7$              ;;LOOP ON THE PRESENT TEST
1392  002250                            6$:;#####END OF CODE FOR THE XOR TESTER#####
1393  002250  032737  000400  177570            BIT     #BIT08,@#SWR    ;;LOOP ON SPEC. TEST?
1394  002256  001407                            BEQ     2$              ;;BR IF NO
1395  002260  013746  177570                    MOV     @#SWR,-(SP)     ;;SET DESIRED TEST NUM. FROM SWR
1396  002264  042716  000200                    BIC     #$SWRMK,(SP)    ;;STRIP AWAY UNDESIRED BITS
1397  002270  122637  001102                    CMPB    (SP)+,$TSTNM    ;;ON THE RIGHT TEST?
1398  002274  001462                            BEQ     $OVER           ;;BR IF YES
1399  002276  105737  001103            2$:     TSTB    $ERFLG          ;;HAS AN ERROR OCCURRED?
1400  002302  001421                            BEQ     3$              ;;BR IF NO
1401  002304  123737  001115  001103            CMPB    $ERMAX,$ERFLG   ;;MAX. ERRORS FOR THIS TEST OCCURRED?
1402  002312  101015                            BHI     3$              ;;BR IF NO
1403  002314  032737  001000  177570            BIT     #BIT09,@#SWR    ;;LOOP ON ERROR?
1404  002322  001404                            BEQ     4$              ;;BR IF NO
1405  002324  013737  001110  001106    7$:     MOV     $LPERR,$LPADR   ;;SET LOOP ADDRESS TO LAST SCOPE
1406  002332  000443                            BR      $OVER
1407  002334  105037  001103            4$:     CLRB    $ERFLG          ;;ZERO THE ERROR FLAG
1408  002340  005037  001204                    CLR     $TIMES          ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
1409  002344  000415                            BR      1$              ;;ESCAPE TO THE NEXT TEST
1410  002346  032737  004000  177570    3$:     BIT     #BIT11,@#SWR    ;;INHIBIT ITERATIONS?
1411  002354  001011                            BNE     1$              ;;BR IF YES
1412  002356  005737  001100                    TST     $PASS           ;;IF FIRST PASS OF PROGRAM
1413  002362  001406                            BEQ     1$              ;;      INHIBIT ITERATIONS
1414  002364  005237  001104                    INC     $ICNT           ;;INCREMENT ITERATION COUNT
1415  002370  023737  001204  001104            CMP     $TIMES,$ICNT    ;;CHECK THE NUMBER OF ITERATIONS MADE
1416  002376  002021                            BGE     $OVER           ;;BR IF MORE ITERATION REQUIRED
1417  002400  012737  000001  001104    1$:     MOV     #1,$ICNT        ;;REINITIALIZE THE ITERATION COUNTER
1418  002406  013737  002456  001204            MOV     $MXCNT,$TIMES   ;;SET NUMBER OF ITERATIONS TO DO
1419  002414  105237  001102            $SVLAD: INCB    $TSTNM          ;;COUNT TEST NUMBERS
1420  002420  011637  001106                    MOV     (SP),$LPADR     ;;SAVE SCOPE LOOP ADDRESS
1421  002424  011637  001110                    MOV     (SP),$LPERR     ;;SAVE ERROR LOOP ADDRESS
1422  002430  005037  001206                    CLR     $ESCAPE         ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
1423  002434  112737  000001  001115            MOVB    #1,$ERMAX       ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
1424  002442  013737  001102  177570    $OVER:  MOV     $TSTNM,@#DISPLAY;;DISPLAY TEST NUMBER
1425  002450  013716  001106                    MOV     $LPADR,(SP)     ;;FUDGE RETURN ADDRESS
1426  002454  000002                            RTI                     ;;FIXES PS
1427  002456  000310                    $MXCNT: 200.                    ;;MAX. NUMBER OF ITERATIONS
1428                                    ;;*****************************************************************
1429
1430                                    .SBTTL  ERROR HANDLER ROUTINE
1431
1432                                    ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
1433                                    ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
```

```
                                        ;*AND GO TO ERTYPE ON ERROR
                                        ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
                                        ;*SW15=1        HALT ON ERROR
                                        ;*              HALT CAN OCCUR BEFORE AND AFTER THE ERROR TYPEOUT
                                        ;*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

002460                                  $ERROR:
002460  113737  001102  001262                 MOVB    $TSTNM,TESTNO   ;SAVE TEST NUMBER FOR ERROR TYPE OUT
002466  005237  001254                         INC     ERRCNT          ;COUNT ALL MULTIPLE ERRORS
002472  010037  001154                         MOV     R0,$REG0        ;SAVE R0 FOR POSSIBLE TYPE OUT
002476  010137  001156                         MOV     R1,$REG1        ;SAVE R1 FOR POSSIBLE TYPE OUT
002502  010237  001160                         MOV     R2,$REG2        ;SAVE R2 FOR POSSIBLE TYPE OUT
002506  010337  001162                         MOV     R3,$REG3        ;SAVE R3 FOR POSSIBLE TYPE OUT
002512  010437  001164                         MOV     R4,$REG4        ;SAVE R4 FOR POSSIBLE TYPE OUT
002516  010537  001166                         MOV     R5,$REG5        ;SAVE R5 FOR POSSIBLE TYPE OUT
002522  105237  001103                 7$:     INCB    $ERFLG          ;;SET THE ERROR FLAG
002526  001775                               BEQ     7$              ;;DON'T LET THE FLAG GO TO ZERO
002530  013737  001102  177570                 MOV     $TSTNM,@#DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
002536  005737  177570                         TST     @#SWR           ;;HALT ON ERROR = 1?
002542  100001                               BPL     8$              ;;BRANCH IF NO
002544  000000                               HALT                    ;;YES--HALT
002546  032737  002000  177570  8$:     BIT     #BIT10,@#SWR    ;;BELL ON ERROR?
002554  001402                               BEQ     1$              ;;NO - SKIP
002556  104400  001210                         TYPE    ,$BELL          ;;RING BELL
002562  005237  001112                 1$:     INC     $ERTTL          ;;COUNT THE NUMBER OF ERRORS
002566  011637  001116                         MOV     (SP),$ERRPC     ;;GET ADDRESS OF ERROR INSTRUCTION
002572  162737  000002  001116                 SUB     #2,$ERRPC
002600  117737  176312  001114                 MOVB    @$ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
002606  032737  020000  177570                 BIT     #BIT13,@#SWR    ;;SKIP TYPEOUT IF SET
002614  001004                               BNE     2$              ;;SKIP TYPEOUTS
002616  004737  002762                         JSR     PC,ERTYPE       ;;GO TO USER ERROR ROUTINE
002622  104400  001215                         TYPE    ,$CRLF
002626  005737  177570                 2$:     TST     @#SWR           ;;HALT ON ERROR
002632  100001                               BPL     9$              ;;SKIP IF CONTINUE
002634  000000                               HALT                    ;;HALT ON ERROR!
002636  022737  023100  000042  9$:     CMP     #$ENDAD,42      ;;ACT-11?
002644  001001                               BNE     3$              ;;BRANCH IF NO
002646  000000                               HALT                    ;;YES
002650  032737  001000  177570  3$:     BIT     #BIT09,@#SWR    ;;LOOP ON ERROR SWITCH SET?
002656  001402                               BEQ     4$              ;;BR IF NO
002660  013716  001110                         MOV     $LPERR,(SP)     ;;FUDGE RETURN FOR LOOPING
002664  005737  001206                 4$:     TST     $ESCAPE         ;;CHECK FOR AN ESCAPE ADDRESS
002670  001402                               BEQ     5$              ;;BR IF NONE
002672  013716  001206                         MOV     $ESCAPE,(SP)    ;;FUDGE RETURN ADDRESS FOR ESCAPE
002676                                  5$:
002676  032737  001000  177570                 BIT     #SW9,SWR        ;ARE YOU LOOPING ON THIS ERROR?
002704  001425                               BEQ     EREXIT          ;BRANCH IF NOT LOOPING ON ERROR
002706  012737  177777  177766                 MOV     #-1,@#CPUERR    ;CLEAR CPU ERROR REGISTER
002714  012737  177777  177744                 MOV     #-1,@#MEMERR    ;CLEAR MEMORY ERROR REGISTER
002722  042737  177776  177572                 BIC     #177776,@#MMR0  ;CLEAR MEMORY MANAGEMENT STATUS REGISTER
002730  012737  177777  005074                 MOV     #-1,TOFLAG      ;INITIALIZE TRAP FLAG
002736  012737  177777  005574                 MOV     #-1,CPFLAG      ;INITIALIZE CP TRAP FLAG
```

```
1490  002744  012737  177777  005706          MOV     #-1,PAFLAG      ;INITIALIZE PARITY TRAP FLAG
1491  002752  012737  177777  006132          MOV     #-1,MMFLAG      ;INITIALIZE MEMORY MANAGEMENT TRAP FLAG
1492  002760  000002                  EREXIT: RTI                     ;RETURN TO TEST
1493
1494
1495                                  .SBTTL  ERROR MESSAGE TYPE OUT ROUTINE
1496
1497
1498                                  ;*      THIS SUBROUTINE IS CALLED BY THE ERROR HANDLER TO TYPE
1499                                  ;*      THE ERROR MESSAGES.  IT PICKS UP THE ITEM BYTE ($ITEMB) NUMBER
1500                                  ;*      AND USES THAT TO INDEX THROUGH THE ERROR TABLE.  THE ERROR
1501                                  ;*      TABLE STARTS AT "$ERRTB" AND HAS FOUR (4) POINTERS FOR EACH
1502                                  ;*      ENTRY, 'EM'  'DH'  'DT'  'DF'.  THE 'EM' POINTS TO THE ERROR
1503                                  ;*      MESSAGE WHICH IS AN ASCIZ STRING.  THE 'DH' POINTS TO THE DATA
1504                                  ;*      HEADER WHICH IS ANOTHER ASCIZ STRING.  THE 'DT' POINTS TO THE
1505                                  ;*      DATA TABLE WHICH IS A GROUP OF WORDS CONTAINING THE ADDRESSES
1506                                  ;*      OF THE DATA TO BY TYPED.  THE FORMAT OF THIS DATA IS
1507                                  ;*      CONTROLLED BY THE 'DF' WHICH IS THE POINTER TO THE DATA FORMAT.
1508                                  ;*      THE DATA FORMAT IS A GROUP OF BYTES WHICH CONTAIN NUMBERS
1509                                  ;*      THAT CORRESPOND TO DIFFERENT TYPING FORMATS.
1510                                  ;*      0       -16 BIT OCTAL FORMAT
1511                                  ;*      1       -DECIMAL FORMAT
1512                                  ;*      2       -22 BIT OCTAL FORMAT.  DATA IS LOWER 16 BITS OF THE
1513                                  ;*               PHYSICAL ADDRESS, UPPER 6 BITS ARE ADJACENT TO LOWER 16
1514                                  ;*      3       -22 BIT OCTAL FORMAT.  DATA IS THE 16 BIT VIRTUAL
1515                                  ;*               ADDRESS IN KERNEL I-SPACE.
1516                                  ;*      4       -18 BIT OCTAL FORMAT.  DATA IS A 16 BIT NUMBER THAT
1517                                  ;*               WILL BE CONVERTED INTO A UNIBUS ADDRESS BY LEFT
1518                                  ;*               SHIFTING IT 6 BITS.
1519  002762  010046                  ERTYPE: MOV     R0,-(KSP)       ;SAVE R0 ON STACK
1520  002764  005000                          CLR     R0              ;CLEAR R0
1521  002766  113700  001114                  MOVB    @#$ITEMB,R0     ;PUT ITEM NUMBER IN R0
1522  002772  001004                          BNE     1$              ;BRANCH IF IT IS NON-ZERO
1523  002774  013746  001116                  MOV     $ERRPC,-(KSP)   ;PUT ERROR PC ON STACK FOR TYPING
1524  003000  104402                          TYPOC                   ;TYPE FAILING PC
1525  003002  000526                          BR      13$             ;GO TO RETURN
1526  003004  005300                  1$:     DEC     R0              ;ADJUST ITEM NUMBER TO BE A POINTER
1527  003006  072027  000003                  ASH     #3,R0           ;LEFT SHIFT ITEM NO. 3 PLACES
1528  003012  100024                          BPL     22$             ;BRANCH IF ITEM #LESS THAN 200
1529  003014  032700  001000                  BIT     #BIT9,R0        ;SEE IF ITEM # WAS 3XX
1530  003020  001415                          BEQ     21$             ;BRANCH IF ITEM # WAS 2XX
1531                                                                  ;AND TYPE ERROR MESSAGE
1532  003022  032737  000200  177570          BIT     #SW7,@#SWR      ;SEE IF SWITCH 7 IS UP
1533  003030  001404                          BEQ     20$             ;BRANCH IF SWITCH IS NOT UP
1534                                                                  ;AND TYPE DATA, ON MULTIPLE ERRORS
1535  003032  062766  000004  000002          ADD     #4,2(KSP)       ;SKIP 'TYPE  ,$CRLF' IF SW 7 IS UP
1536                                                                  ;INHIBIT MULTIPLE ERROR TYPEOUTS
1537  003040  000507                          BR      13$             ;BRANCH TO EXIT
1538  003042  042700  177000          20$:    BIC     #177000,R0      ;CLEAR UPPER BYTE OF R0
1539  003046  062700  002054                  ADD     #ER200+4,R0     ;POINT TO DATA TABLE ENTRY
1540  003052  000424                          BR      5$              ;GO TYPE DATA TABLE
1541  003054  042700  177000          21$:    BIC     #177000,R0      ;CLEAR UPPER BYTE OF R0
1542  003060  062700  000520                  ADD     #<ER200-$ERRTB>,R0 ;ADD DIFFERENCE BETWEEN
1543                                                                  ;ITEM 1 AND ITEM 201
1544                                  ;;                      GET POINTER TO ERROR MESSAGE AND TYPE IT
1545                                  ;;                      IF THE POINTER IS NOT ZERO
```

```
1546  003064  062700  001330          22$:    ADD     #$ERRTB,R0      ;ADD BASE OF ERROR TABLE
1547  003070  012037  003100                  MOV     (R0)+,2$        ;PUT MESSAGE POINTER IN TYPE STATEMENT
1548  003074  001404                          BEQ     3$              ;BRANCH IF NO ERROR MESSAGE
1549  003076  104400                          TYPE                    ;TYPE ERROR MESSAGE
1550  003100  000000                  2$:     .WORD   0               ;POINTER TO ERROR MESSAGE
1551  003102  104400  001215                  TYPE    ,$CRLF          ;TYPE CRLF
1552                                  ;;                      GET THE POINTER TO THE DATA HEADER AND
1553                                  ;;                      TYPE IT IF THE POINTER IS NOT ZERO
1554  003106  012037  003116          3$:     MOV     (R0)+,4$        ;PUT HEADER POINTER IN TYPE STATEMENT
1555  003112  001404                          BEQ     5$              ;BRANCH IF NO DATA HEADER
1556  003114  104400                          TYPE                    ;TYPE THE DATA HEADER
1557  003116  000000                  4$:     .WORD   0               ;POINTER TO DATA HEADER
1558  003120  104400  001215                  TYPE    ,$CRLF          ;TYPE CRLF
1559                                  ;;                      THIS IS THE START OF THE DATA OUTPUT IF THE
1560                                  ;;                      DATA POINTER IS NOT ZERO.  R0 POINTS TO THE
1561                                  ;;                      DATA FORMAT, R1 POINTS TO THE ADDRESS OF
1562                                  ;;                      THE DATA WORDS.
1563  003124  010146                  5$:     MOV     R1,-(KSP)       ;SAVE R1 ON THE STACK
1564  003126  012001                          MOV     (R0)+,R1        ;PUT DATA TABLE POINTER IN R1
1565  003130  001452                          BEQ     12$             ;BRANCH IF NO DATA TABLE
1566  003132  012000                          MOV     (R0)+,R0        ;PICK UP DATA FORMAT POINTER
1567  003134  105710                  6$:     TSTB    (R0)            ;IS THIS WORD OCTAL
1568  003136  001003                          BNE     7$              ;BRANCH IF NOT 16-BIT OCTAL
1569                                  ;;                      THIS IS 16 BIT OCTAL FORMAT (DF = 0)
1570  003140  013146                          MOV     @(R1)+,-(KSP)   ;PUT WORD ON STACK FOR TYPING
1571  003142  104402                          TYPOC                   ;TYPE THE WORD ON STACK AS 16 BIT OCTAL
1572  003144  000436                          BR      11$             ;GET READY FOR NEXT WORD
1573  003146  122710  000001          7$:     CMPB    #1,(R0)         ;IS THE WORD DECIMAL
1574  003152  001003                          BNE     8$              ;BRANCH IF NOT DECIMAL
1575                                  ;;                      THIS IS DECIMAL FORMAT (DF = 1)
1576  003154  013146                          MOV     @(R1)+,-(KSP)   ;PUT WORD ON STACK FOR TYPING
1577  003156  104410                          TYPDS                   ;TYPE THE WORD ON STACK AS DECIMAL
1578  003160  000430                          BR      11$             ;GET READY FOR NEXT WORD
1579  003162  122710  000002          8$:     CMPB    #2,(R0)         ;IS WORD 22-BIT PHYSICAL ADDRESS
1580  003166  001012                          BNE     9$              ;BRANCH IF NOT 22-BIT PHYSICAL ADDR
1581                                  ;;                      THIS IS 22-BIT PHYSICAL FORMAT (DF = 2)
1582  003170  012146                          MOV     (R1)+,-(KSP)    ;PUT ADDR OF LOW WORD ON STACK
1583  003172  004737  004670                  JSR     PC,$DB2O        ;CONVERT NUMBER TO OCTAL ASCIZ
1584  003176  062716  000003                  ADD     #3,(KSP)                ;ONLY WANT 8 DIGITS
1585  003202  012637  003210                  MOV     (KSP)+,30$      ;PUT POINTER AFTER 'TYPE' CALL
1586  003206  104400                          TYPE                    ;TYPE ASCIZ STRING
1587  003210  000000                  30$:    .WORD   0               ;WORD HOLDS POINTER TO ASCIZ STRING
1588  003212  000413                          BR      11$             ;GET READY FOR NEXT WORD
1589  003214  122710  000003          9$:     CMPB    #3,(R0)         ;IS THIS A 16-BIT VIRTUAL ADDRESS
1590  003220  001004                          BNE     10$             ;BRANCH IF NOT 16-BIT VIRT. ADDR.
1591                                  ;;                      THIS IS 22-BIT VIRTUAL ADDRESS FORMAT
1592                                  ;;                      KERNEL I-SPACE ASSUMED.  (DF = 3)
1593  003222  013146                          MOV     @(R1)+,-(KSP)   ;PUT 16-BIT VIRTUAL ADDR ON STACK
1594  003224  004737  003270                  JSR     PC,TYPVAD       ;GO TYPE 22-BIT ADDRESS FROM 16-BIT V.A.
1595  003230  000404                          BR      11$             ;GET READY FOR NEXT WORD
1596                                  ;;                      THIS IS FORMAT 4.  DATA WORD IS A UNIBUS ADDRESS
1597                                  ;;                      OUTPUT WILL BE 18-BITS WORD LEFT SHIFTED 6.
1598  003232  013146                  10$:    MOV     @(R1)+,-(KSP)   ;PUSH 16-BIT UNIBUS ADDRESS ON STACK
1599  003234  004737  003376                  JSR     PC,UBADDR       ;CONVERT TO 18-BIT UNIBUS ADDR AND TYPE
1600  003240  000400                          BR      11$             ;GET READY FOR NEXT WORD
1601  003242  005200                  11$:    INC     R0              ;POINT TO NEXT FORMAT BYTE
```

```
1602  003244  104400  003264                  TYPE    ,32$            ;TYPE TWO SPACES
1603  003250  005711                          TST     (R1)            ;IS THERE ANOTHER WORD?
1604  003252  001401                          BEQ     12$             ;BRANCH IF ALL DONE
1605  003254  000727                          BR      6$              ;GO BACK FOR NEXT NUMBER
1606  003256  012601                  12$:    MOV     (KSP)+,R1       ;RESTORE R1
1607  003260  012600                  13$:    MOV     (KSP)+,R0       ;RESTORE R0
1608  003262  000207                          RTS     PC              ;RETURN TO ERROR ROUTINE
1609  003264  020040      000         32$:    .ASCIZ  ?  ?            ;TWO SPACES
1610          003270                          .EVEN
1611
1612
1613                                  .SBTTL  CONVERT 16-BIT VIRTUAL ADDRESS TO 22-BIT PHYSICAL ADDRESS
1614                                  ;*
1615                                  ;*      THIS ROUTINE IS CALLED BY A 'JSR PC' AFTER THE VIRTUAL ADDRESS
1616                                  ;*      IS PUSHED ON THE KERNEL STACK.  THE V.A. IS THEN LOADED INTO
1617                                  ;*      R1 AND THE UPPER 3 BITS ARE SHIFTED INTO R0 TO SELECT THE
1618                                  ;*      CORRECT KERNEL I-SPACE PAR.  THE LOWER 12 BITS OF THE VIRTUAL
1619                                  ;*      ADDRESS ARE ADDED TO THE PAR AS THEY ARE BY MEMORY MANAGEMENT
1620                                  ;*      AND THE PHYSICAL ADDRESS IS SAVED IN MEMORY TO BE CONVERTED
1621                                  ;*      TO ASCIZ AND TYPED.
1622                                  ;*
1623  003270  104412                  TYPVAD: SAVREG                  ;SAVE ALL REGISTERS
1624  003272  016601  000002                  MOV     2(KSP),R1       ;PUT VIRTUAL ADDR IN R1
1625  003276  005000                          CLR     R0              ;CLEAR R0 FOR CALCULATIONS
1626  003300  073027  000003                  ASHC    #3,R0           ;LEFT SHIFT R0,R1 3 PLACES
1627  003304  006300                          ASL     R0              ;LEFT SHIFT R0 ONE MORE PLACE
1628  003306  006001                          ROR     R1              ;RIGHT SHIFT R1 SO OFFSET IS CORRECT
1629  003310  006001                          ROR     R1              ;RIGHT SHIFT R1
1630  003312  006001                          ROR     R1              ;RIGHT SHIFT R1
1631  003314  062700  172340                  ADD     #KIPAR0,R0      ;FORM DESIRED PAR ADDR IN R0
1632  003320  011003                          MOV     (R0),R3         ;PUT CONTENTS OF PAR IN R3
1633  003322  005002                          CLR     R2              ;CLEAR R2 FOR PHYSICAL ADDR CALCULATIONS
1634  003324  073227  000006                  ASHC    #6,R2           ;LEFT SHIFT <R2,R3> 6 PLACES
1635  003330  060103                          ADD     R1,R3           ;ADD OFFSET IN R1 TO BASE IN R3
1636  003332  005502                          ADC     R2              ;ADD ANY POSSIBLE CARRY TO UPPER 6 BITS
1637  003334  010237  001222                  MOV     R2,PADRSH       ;PUT UPPER 6 BITS OF ADDR IN CORE
1638  003340  010337  001220                  MOV     R3,PADRSL       ;PUT LOWER 16 BITS OF ADDR IN CORE
1639  003344  012746  001220                  MOV     #PADRSL,-(KSP)  ;PUT POINTER TO LOWER 16 BITS ON STACK
1640  003350  004737  004670                  JSR     PC,$DB2O        ;CONVERT NUMBER TO OCTAL ASCIZ
1641  003354  062716  000003                  ADD     #3,(KSP)        ;ONLY TYPE 8 DIGITS
1642  003360  012637  003366                  MOV     (KSP)+,3$       ;PUT POINTER AFTER TYPE INST
1643  003364  104400                          TYPE                    ;TYPE THE 22-BIT VIRTUAL ADDRESS
1644  003366  000000              3$:         .WORD   0               ;THIS WORD HOLDS THE POINTER TO
1645                                                                  ;THE ASCIZ STRING
1646  003370  104414                          RESREG                  ;RESTORE ALL THE REGISTERS
1647  003372  012616                          MOV     (KSP)+,(KSP)    ;LEAVE ONLY RETURN ADDR ON STACK
1648  003374  000207                          RTS     PC              ;RETURN TO ERROR HANDLER
1649
1650
1651                                  ;*THIS SUBROUTINE IS USED TO CONVERT THE A WORD PUSHED
1652                                  ;*ON THE STACK INTO A UNIBUS ADDRESS AND TYPE IT AS A
1653                                  ;*6 DIGIT NUMBER.  IT USES R1 & R0 AND LEAVES
1654                                  ;*ALL OTHER REGISTERS UNCHANGED.
1655  003376  016601  000002          UBADDR: MOV     2(KSP),R1       ;LOAD 16 BIT ADDRESS INTO R1
1656  003402  005000                          CLR     R0              ;CLEAR R0 FOR CALCULATIONS
1657  003404  073027  000006                  ASHC    #6,R0           ;LEFT SHIFT <R0:R1> 6 PLACES
```

```
1658   003410  010137  001220                  MOV     R1,PADRSL       ;PUT LOWER 16 BITS IN PADRSL
1659   003414  010037  001222                  MOV     R0,PADRSH       ;PUT UPPER 6 BITS IN PADRSH
1660   003420  012746  001220                  MOV     #PADRSL,-(KSP)  ;PUSH POINTER TO WORDS ON STACK
1661   003424  004737  004670                  JSR     PC,$DB2O        ;JUMP TO CONVERT ROUTINE
1662   003430  062716  000005                  ADD     #5,(KSP)        ;ONLY USE LOWER 6 CHARS.
1663   003434  012637  003442                  MOV     (KSP)+,3$       ;PUT POINTER AFTER TYPE CALL.
1664   003440  104400                          TYPE
1665   003442  000000                  3$:     .WORD   0               ;HOLDS POINTER TO FIRST CHAR.
1666   003444  012616                          MOV     (KSP)+,(KSP)    ;LEAVE ONLY RETURN ADDRESS ON STACK
1667   003446  000207                          RTS     PC              ;RETURN TO ERROR TYPE ROUTINE.
1668
1669
1670                                  ;;*****************************************************************
1671
1672                                  .SBTTL  SAVE AND RESTORE R0-R5 ROUTINES
1673
1674                                  ;*SAVE R0-R5
1675                                  ;*CALL:
1676                                  ;*      SAVREG
1677                                  ;*UPON RETURN FROM $SAVREG THE STACK WILL LOOK LIKE:
1678                                  ;*
1679                                  ;*TOP---(+16)
1680                                  ;* +2---(+18)
1681                                  ;* +4---R5
1682                                  ;* +6---R4
1683                                  ;* +8---R3
1684                                  ;*+10---R2
1685                                  ;*+12---R1
1686                                  ;*+14---R0
1687
1688   003450                         $SAVREG:
1689   003450  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
1690   003452  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
1691   003454  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
1692   003456  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
1693   003460  010446                          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
1694   003462  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
1695   003464  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PS OF MAIN FLOW
1696   003470  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PC OF MAIN FLOW
1697   003474  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PS OF CALL
1698   003500  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PC OF CALL
1699   003504  000002                          RTI
1700
1701                                  ;*RESTORE R0-R5
1702                                  ;*CALL:
1703                                  ;*      RESREG
1704   003506                         $RESREG:
1705   003506  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PC OF CALL
1706   003512  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PS OF CALL
1707   003516  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PC OF MAIN FLOW
1708   003522  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PS OF MAIN FLOW
1709   003526  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
1710   003530  012604                          MOV     (SP)+,R4        ;;POP STACK INTO R4
1711   003532  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
1712   003534  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
1713   003536  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
```

```
1714  003540  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
1715  003542  000002                          RTI
1716                                  ;;*****************************************************************
1717
1718                                  .SBTTL  TYPE ROUTINE
1719
1720                                  ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
1721                                  ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
1722                                  ;*NOTE1:        $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
1723                                  ;*NOTE2:        $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
1724                                  ;*NOTE3:        $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
1725                                  ;*
1726                                  ;*CALL:
1727                                  ;*1) USING A TRAP INSTRUCTION
1728                                  ;*      TYPE    ,MESADR         ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
1729                                  ;*OR
1730                                  ;*      TYPE
1731                                  ;*      MESADR
1732                                  ;*
1733                                  ;*2) USING A JSR INSTRUCTION
1734                                  ;*      MOV     PS,-(SP)        ;;PUSH PROCESSOR STATUS WORD ON THE STACK
1735                                  ;*      JSR     PC,$TYPE        ;;CALL TYPE ROUTINE
1736                                  ;*      MESADDR                 ;;FIRST ADRESS OF MESSAGE
1737
1738  003544  105737  001151          $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
1739  003550  100002                          BPL     1$              ;;BR IF YES
1740  003552  000000                          HALT                    ;;HALT HERE IF NO TERMINAL
1741  003554  000407                          BR      3$              ;;LEAVE
1742  003556  010046                  1$:     MOV     R0,-(SP)        ;;SAVE R0
1743  003560  017600  000002                  MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
1744  003564  112046                  2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
1745  003566  001005                          BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
1746  003570  005726                          TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
1747  003572  012600                          MOV     (SP)+,R0        ;;RESTORE R0
1748  003574  062716  000002          3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
1749  003600  000002                          RTI                     ;;RETURN
1750  003602  122716  000011          4$:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
1751  003606  001424                          BEQ     8$
1752  003610  122716  000200                  CMPB    #CRLF,(SP)      ;;BRANCH IF NOT
1753  003614  001004                          BNE     5$
1754  003616  005726                          TST     (SP)+           ;;POP  <CR><LF> EQUIV
1755  003620  104400  001215                  TYPE    ,$CRLF
1756  003624  000757                          BR      2$              ;;GET NEXT CHARACTER
1757  003626  004737  003704          5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
1758  003632  123726  001150          6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
1759  003636  001352                          BNE     2$              ;;IF NO GO GET NEXT CHAR.
1760  003640  013746  001146                  MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
1761                                                                  ;;AND THE NULL CHAR.
1762  003644  105366  000001          7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
1763  003650  002770                          BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
1764  003652  004737  003704                  JSR     PC,$TYPEC       ;;GO TYPE A NULL
1765  003656  000772                          BR      7$              ;;LOOP
1766
1767                                  ;;HORIZONTAL TAB PROCESSOR
1768
1769  003660  112716  000040          8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
```

```
1770  003664  004737  003704          9$:     JSR     PC,$TYPEC       ;;TYPE A SPACE
1771  003670  132737  000007  003750          BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
1772  003676  001372                          BNE     9$              ;;TAB STOP
1773  003700  005726                          TST     (SP)+           ;;POP SPACE OFF STACK
1774  003702  000730                          BR      2$              ;;GET NEXT CHARACTER
1775  003704  105777  175232          $TYPEC: TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
1776  003710  100375                          BPL     $TYPEC
1777  003712  116677  000002  175224        MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
1778  003720  122766  000015  000002        CMPB    #CR,2(SP)       ;;BRANCH IF
1779  003726  001003                          BNE     1$              ;;NOT <CR>
1780  003730  105037  003750                CLRB    $CHARCNT        ;;
1781  003734  000406                          BR      $TYPEX          ;;EXIT
1782  003736  122766  000012  000002  1$:     CMPB    #LF,2(SP)       ;;BRANCH IF
1783  003744  001402                          BEQ     $TYPEX          ;;<LF>
1784  003746  105227                          INCB    (PC)+           ;;INC SPACE
1785  003750  000000          $CHARCNT:.WORD  0               ;;COUNT
1786  003752  000207          $TYPEX: RTS     PC
1787
1788                          ;;**************************************************************
1789
1790                          .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
1791
1792                          ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
1793                          ;*OCTAL (ASCII) NUMBER AND TYPE IT.
1794                          ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
1795                          ;*CALL:
1796                          ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
1797                          ;*      TYPOS                   ;;CALL FOR TYPEOUT
1798                          ;*      .BYTE   N               ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
1799                          ;*      .BYTE   M               ;;M=1 OR 0
1800                          ;*                                      ;;1=TYPE LEADING ZEROS
1801                          ;*                                      ;;0=SUPPRESS LEADING ZEROS
1802                          ;*
1803                          ;*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
1804                          ;*$TYPOS OR $TYPOC
1805                          ;*CALL:
1806                          ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
1807                          ;*      TYPON                   ;;CALL FOR TYPEOUT
1808                          ;*
1809                          ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
1810                          ;*CALL:
1811                          ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
1812                          ;*      TYPOC                   ;;CALL FOR TYPEOUT
1813
1814  003754  017646  000000          $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
1815  003760  116637  000001  004177        MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
1816  003766  112637  004201                MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
1817  003772  062716  000002                ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
1818  003776  000406                        BR      $TYPON
1819  004000  112737  000001  004177  $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
1820  004006  112737  000006  004201        MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
1821  004014  112737  000005  004176  $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
1822  004022  010346                        MOV     R3,-(SP)        ;;SAVE R3
1823  004024  010446                        MOV     R4,-(SP)        ;;SAVE R4
1824  004026  010546                        MOV     R5,-(SP)        ;;SAVE R5
1825  004030  113704  004201                MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
```

```
1826  004034  005404                          NEG     R4
1827  004036  062704  000006                  ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
1828  004042  110437  004200                  MOVB    R4,$OMODE       ;;SAVE IT FOR USE
1829  004046  113704  004177                  MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
1830  004052  016605  000012                  MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
1831  004056  005003                          CLR     R3              ;;CLEAR THE OUTPUT WORD
1832  004060  006105                  1$:     ROL     R5              ;;ROTATE MSB INTO "C"
1833  004062  000404                          BR      3$              ;;GO DO MSB
1834  004064  006105                  2$:     ROL     R5              ;;FORM THIS DIGIT
1835  004066  006105                          ROL     R5
1836  004070  006105                          ROL     R5
1837  004072  010503                          MOV     R5,R3
1838  004074  006103                  3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
1839  004076  105337  004200                  DECB    $OMODE          ;;TYPE THIS DIGIT?
1840  004102  100016                          BPL     7$              ;;BR IF NO
1841  004104  042703  177770                  BIC     #177770,R3      ;;GET RID OF JUNK
1842  004110  001002                          BNE     4$              ;;TEST FOR 0
1843  004112  005704                          TST     R4              ;;SUPPRESS THIS 0?
1844  004114  001403                          BEQ     5$              ;;BR IF YES
1845  004116  005204                  4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
1846  004120  052703  000060                  BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
1847  004124  052703  000040          5$:     BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
1848  004130  110337  004174                  MOVB    R3,8$           ;;SAVE FOR TYPING
1849  004134  104400  004174                  TYPE    ,8$             ;;GO TYPE THIS DIGIT
1850  004140  105337  004176          7$:     DECB    $OCNT           ;;COUNT BY 1
1851  004144  003347                          BGT     2$              ;;BR IF MORE TO DO
1852  004146  002402                          BLT     6$              ;;BR IF DONE
1853  004150  005204                          INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
1854  004152  000744                          BR      2$              ;;GO DO THE LAST DIGIT
1855  004154  012605                  6$:     MOV     (SP)+,R5        ;;RESTORE R5
1856  004156  012604                          MOV     (SP)+,R4        ;;RESTORE R4
1857  004160  012603                          MOV     (SP)+,R3        ;;RESTORE R3
1858  004162  016666  000002  000004          MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
1859  004170  012616                          MOV     (SP)+,(SP)
1860  004172  000002                          RTI                     ;;RETURN
1861  004174     000                  8$:     .BYTE   0               ;;STORAGE FOR ASCII DIGIT
1862  004175     000                          .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
1863  004176     000                  $OCNT:  .BYTE   0               ;;OCTAL DIGIT COUNTER
1864  004177     000                  $OFILL: .BYTE   0               ;;ZERO FILL SWITCH
1865  004200  000000                  $OMODE: .WORD   0               ;;NUMBER OF DIGITS TO TYPE
1866                                  ;;**************************************************************
1867
1868                                  .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
1869
1870                                  ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
1871                                  ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
1872                                  ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
1873                                  ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
1874                                  ;*REPLACED WITH SPACES.
1875                                  ;*CALL:
1876                                  ;*      MOV     NUM,-(SP)       ;;PUT THE BINARY NUMBER ON THE STACK
1877                                  ;*      TYPDS                   ;;GO TO THE ROUTINE
1878
1879  004202                          $TYPDS:
1880  004202  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
1881  004204  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
```

```
1882   004206   010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
1883   004210   010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
1884   004212   010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
1885   004214   012746   020200                 MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
1886   004220   016605   000020                 MOV     20(SP),R5       ;;GET THE INPUT NUMBER
1887   004224   100004                          BPL     1$              ;;BR IF INPUT IS POS.
1888   004226   005405                          NEG     R5              ;;MAKE THE BINARY NUMBER POS.
1889   004230   112766   000055   000001        MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
1890   004236   005000                  1$:     CLR     R0              ;;ZERO THE CONSTANTS INDEX
1891   004240   012703   004416                 MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
1892   004244   112723   000040                 MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
1893   004250   005002                  2$:     CLR     R2              ;;CLEAR THE BCD NUMBER
1894   004252   016001   004406                 MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
1895   004256   160105                  3$:     SUB     R1,R5           ;;FORM THIS BCD DIGIT
1896   004260   002402                          BLT     4$              ;;BR IF DONE
1897   004262   005202                          INC     R2              ;;INCREASE THE BCD DIGIT BY 1
1898   004264   000774                          BR      3$
1899   004266   060105                  4$:     ADD     R1,R5           ;;ADD BACK THE CONSTANT
1900   004270   005702                          TST     R2              ;;CHECK IF BCD DIGIT=0
1901   004272   001002                          BNE     5$              ;;FALL THROUGH IF 0
1902   004274   105716                          TSTB    (SP)            ;;STILL DOING LEADING 0'S?
1903   004276   100407                          BMI     7$              ;;BR IF YES
1904   004300   106316                  5$:     ASLB    (SP)            ;;MSD?
1905   004302   103003                          BCC     6$              ;;BR IF NO
1906   004304   116663   000001   177777        MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
1907   004312   052702   000060          6$:     BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
1908   004316   052702   000040          7$:     BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
1909   004322   110223                          MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
1910   004324   005720                          TST     (R0)+           ;;JUST INCREMENTING
1911   004326   020027   000010                 CMP     R0,#10          ;;CHECK THE TABLE INDEX
1912   004332   002746                          BLT     2$              ;;GO DO THE NEXT DIGIT
1913   004334   003002                          BGT     8$              ;;GO TO EXIT
1914   004336   010502                          MOV     R5,R2           ;;GET THE LSD
1915   004340   000764                          BR      6$              ;;GO CHANGE TO ASCII
1916   004342   105726                  8$:     TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
1917   004344   100003                          BPL     9$              ;;BR IF NO
1918   004346   116663   177777   177776        MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
1919   004354   105013                  9$:     CLRB    (R3)            ;;SET THE TERMINATOR
1920   004356   012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
1921   004360   012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
1922   004362   012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
1923   004364   012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
1924   004366   012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
1925   004370   104400   004416                 TYPE    ,$DBLK          ;;NOW TYPE THE NUMBER
1926   004374   016666   000002   000004        MOV     2(SP),4(SP)     ;;ADJUST THE STACK
1927   004402   012616                          MOV     (SP)+,(SP)
1928   004404   000002                          RTI                     ;;RETURN TO USER
1929   004406   023420                  $DTBL:  10000.
1930   004410   001750                          1000.
1931   004412   000144                          100.
1932   004414   000012                          10.
1933   004416   000004                  $DBLK:  .BLKW   4
1934                                   ;;**************************************************************************
1935
1936                                   .SBTTL  TRAP DECODER
1937
```

```
                                ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
                                ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
                                ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
                                ;*GO TO THAT ROUTINE.

004426  010046                  $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
004430  016600  000002                  MOV     2(SP),R0        ;;GET TRAP ADDRESS
004434  005740                          TST     -(R0)           ;;BACKUP BY 2
004436  111000                          MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
004440  016000  004446                  MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
004444  000200                          RTS     R0              ;;GO TO ROUTINE


                                .SBTTL  TRAP TABLE

                                ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
                                ;*BY THE "TRAP" INSTRUCTION.

                                ;       ROUTINE
                                ;       -------
004446                          $TRPAD:
004446  003544                          $TYPE   ;;CALL=TYPE     TRAP+0(104400)  TTY TYPEOUT ROUTINE
004450  004000                          $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
004452  003754                          $TYPOS  ;;CALL=TYPOS    TRAP+4(104404)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
004454  004014                          $TYPON  ;;CALL=TYPON    TRAP+6(104406)  TYPE OCTAL NUMBER (AS PER LAST CALL)
004456  004202                          $TYPDS  ;;CALL=TYPDS    TRAP+10(104410) TYPE DECIMAL NUMBER (WITH SIGN)
004460  003450                          $SAVREG ;;CALL=SAVREG   TRAP+12(104412) SAVE R0-R5 ROUTINE
004462  003506                          $RESREG ;;CALL=RESREG   TRAP+14(104414) RESTORE R0-R5 ROUTINE
004464  005010                          TBITOFF ;;CALL=TBITO    TRAP+16(104416) THIS WILL TURN OFF T BIT TRAPPING
004466  005036                          TBITRESTORE     ;;CALL=TBITR    TRAP+20(104420) THIS WILL RETURN THE T BIT TO PR

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

                                .SBTTL  POWER DOWN AND UP ROUTINES

                                ;POWER DOWN ROUTINE
004470  012737  004616  000024  $PWRDN: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
004476  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
004504  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
004506  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
004510  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
004512  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
004514  010446                          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
004516  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
004520  010637  004622                  MOV     SP,$SAVR6       ;;SAVE SP
004524  012737  004536  000024          MOV     #$PWRUP,@#PWRVEC ;;SET UP VECTOR
004532  000000                          HALT
004534  000776                          BR      .-2             ;;HANG UP

                                ;POWER UP ROUTINE
004536  013706  004622          $PWRUP: MOV     $SAVR6,SP       ;;GET SP
004542  005037  004622                  CLR     $SAVR6          ;;WAIT LOOP FOR THE TTY
004546  005237  004622          1$:     INC     $SAVR6          ;;WAIT FOR THE INC
004552  001375                          BNE     1$              ;;OF  WORD
004554  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
004556  012604                          MOV     (SP)+,R4        ;;POP STACK INTO R4
```

```
1994  004560  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
1995  004562  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
1996  004564  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
1997  004566  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
1998  004570  012737  004470  000024          MOV     #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
1999  004576  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
2000  004604  104400                          TYPE                    ;;REPORT THE POWER FAILURE
2001  004606  004624                  $PWRMG: .WORD   PWRMSG           ;;POWER FAIL MESSAGE POINTER
2002  004610  012716                          MOV     (PC)+,(SP)      ;;RESTART AT START
2003  004612  010000                  $PWRAD: .WORD   START           ;;RESTART ADDRESS
2004  004614  000002                          RTI
2005  004616  000000                  $ILLUP: HALT                    ;;THE POWER UP SEQUENCE WAS STARTED
2006  004620  000776                          BR      .-2             ;; BEFORE THE POWER DOWN WAS COMPLETE
2007  004622  000000                  $SAVR6: 0                       ;;PUT THE SP HERE
2008  004624  006412  047520  042527  PWRMSG: .ASCIZ  <12><15>?POWER FAILURE, RESTARTING PROGRAM?
2009  004632  020122  040506  046111
2010  004640  051125  026105  051040
2011  004646  051505  040524  052122
2012  004654  047111  020107  051120
2013  004662  043517  040522  000115
2014                                  .EVEN
2015
2016                          ;;*****************************************************************
2017
2018                          .SBTTL  DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE
2019
2020                          ;*THIS ROUTINE WILL CONVERT A 32-BIT UNSIGNED BINARY NUMBER TO AN
2021                          ;*UNSIGNED OCTAL ASCIZ NUMBER.
2022                          ;*CALL
2023                          ;*      MOV     #PNTR,-(SP)     ;;POINTER TO LOW WORD OF BINARY NUMBER
2024                          ;*      JSR     PC,@#$DB20      ;;CALL THE ROUTINE
2025                          ;*      RETURN                  ;;THE ADDRESS OF THE FIRST ASCIZ CHAR. IS ON THE STACK
2026
2027
2028  004670  104412          $DB20:  SAVREG                  ;;SAVE ALL REGISTERS
2029  004672  016601  000002          MOV     2(SP),R1        ;;PICKUP THE POINTER TO LOW WORD
2030  004676  012705  005007          MOV     #$OCTVL+13.,R5  ;;POINTER TO DATA TABLE
2031  004702  012704  000014          MOV     #12.,R4         ;;DO ELEVEN CHARACTERS
2032  004706  012703  177770          MOV     #^C7,R3         ;;MASK
2033  004712  012100                  MOV     (R1)+,R0        ;;LOWER WORD
2034  004714  012101                  MOV     (R1)+,R1        ;;HIGH WORD
2035  004716  005002                  CLR     R2              ;;TERMINATOR
2036  004720  110245          1$:     MOVB    R2,-(R5)        ;;PUT CHARACTER IN DATA TABLE
2037  004722  010002                  MOV     R0,R2           ;;GET THIS DIGIT
2038  004724  005304                  DEC     R4              ;;COUNT THIS CHARACTER
2039  004726  003007                  BGT     3$              ;;BR IF NOT THE LAST DIGIT
2040  004730  001405                  BEQ     2$              ;;BR IF IT IS THE LAST DIGIT
2041  004732  005205                  INC     R5              ;;ALL DIGITS DONE-ADJUST POINTER FOR FIRST
2042  004734  010566  000002          MOV     R5,2(SP)        ;;ASCIZ CHAR. & PUT IT ON THE STACK
2043  004740  104414                  RESREG                  ;;RESTORE ALL REGISTERS
2044  004742  000207                  RTS     PC              ;;RETURN TO USER
2045  004744  006203          2$:     ASR     R3              ;;POSITION THE MASK FOR THE LAST DIGIT
2046  004746  006001          3$:     ROR     R1              ;;POSITION THE BINARY NUMBER FOR
2047  004750  006000                  ROR     R0              ;;      THE NEXT OCTAL DIGIT
2048  004752  006001                  ROR     R1
2049  004754  006000                  ROR     R0
```

```
2050  004756  006001                          ROR     R1
2051  004760  006000                          ROR     R0
2052  004762  040302                          BIC     R3,R2           ;;MASK OUT ALL JUNK
2053  004764  062702  000060                  ADD     #'0,R2          ;;MAKE THIS CHAR. ASCII
2054  004770  000753                          BR      1$              ;;GO PUT IT IN THE DATA TABLE
2055  004772  000016                  $OCTVL: .BLKB   14.             ;;RESERVE DATA TABLE
2056
2057
2058                                  .SBTTL  ************** SUBROUTINES UNIQUE TO THIS PROGRAM ***************
2059
2060
2061
2062                                  .SBTTL  TURN OFF AND SAVE T-BIT
2063                                  ;;**********************************************************************
2064                                  ;*
2065                                  ;*      THIS SUBROUTINE IS REACHED BY THE TRAP CALL 'TBITO'.  IT IS
2066                                  ;*      USED TO TURN OFF THE T-BIT IF IT IS ON.  THE PROCESSOR STATUS
2067                                  ;*      IS SAVED IN "OLDPSW" SO THAT THE T-BIT CAN BE RESTORED TO ITS
2068                                  ;*      PREVIOUS STATUS WHEN CONDITIONS WARRANT.
2069                                  ;*
2070                                  ;;**********************************************************************
2071                                  .EQUIV  BIT4,TBIT       ;T-BIT IS BIT04 IN PROC. STATUS
2072  005010                  TBITOFF:
2073  005010  032766  000020  000002          BIT     #TBIT,2(KSP)    ;IS THE T-BIT ON?
2074  005016  001406                          BEQ     1$              ;BRANCH TO EXIT IF IT IS NOT ON
2075  005020  016637  000002  001310          MOV     2(KSP),OLDPSW   ;SAVE OLD PSW FOR RESTORING T BIT
2076  005026  042766  000020  000002          BIC     #TBIT,2(KSP)    ;CLEAR T BIT IF IT IS ON
2077  005034  000006                  1$:     RTT                     ;RETURN TO PROGRAM
2078
2079
2080
2081                                  .SBTTL  RESTORE T-BIT TO ITS PREVIOUS CONDITION
2082                                  ;;**********************************************************************
2083                                  ;*
2084                                  ;*      THIS SUBROUTINE CAN BE REACHED BY THE TRAP CALL 'TBITR'.  IT IS
2085                                  ;*      USED TO RESTORE THE T-BIT AFTER A PARTICULAR TEST THAT CANNOT
2086                                  ;*      BE RUN WITH THE T-BIT ON.  IT USES THE PROCESSOR STATUS STORED
2087                                  ;*      IN "OLDPSW" BY 'TBITO', REPLACES THE PS ON THE STACK WITH IT
2088                                  ;*      AND DOES AN 'RTT'.
2089                                  ;*
2090                                  ;;**********************************************************************
2091  005036                  TBITRESTORE:
2092  005036  013766  001310  000002          MOV     OLDPSW,2(KSP)   ;PUT OLD PSW ON STACK
2093  005044  042737  000020  001310          BIC     #TBIT,OLDPSW    ;CLEAR T-BIT IN "OLDPSW" SO THAT
2094                                                                  ;IT WON'T BE TURNED ON BY ACCIDENT
2095  005052  000006                          RTT                     ;RETURN TO PROGRAM AND INHIBIT
2096                                                                  ;T BIT TRAP AFTER THIS INSTRUCTION.
2097
2098
2099
2100                                  .SBTTL  SUBROUTINE TO CLEAR ALL OF THE MAP REGISTERS
2101                                  ;;**********************************************************************
2102                                  ;*
2103                                  ;*      THIS SUBROUTINE CLEARS ALL OF THE MAP REGISTERS BY LOADING
2104                                  ;*      THE ADDRESS OF MAPLOO INTO R3 AND THEN CLEARING THE
2105                                  ;*      REGISTER POINTED TO BY R3 UNTIL R3 POINTS ABOVE MAPH37.
```

```
                                        ;*
                                        ;;*************************************************************************
005054  012703  170200          CLRMAP: MOV     #MAPLO,R3       ;PUT FIRST MAP ADDR IN R3
005060  005023                  1$:     CLR     (R3)+           ;CLEAR MAP REGS
005062  022703  170376                  CMP     #MAPH37,R3      ;SEE IF LAST ADDR IS IN R3
005066  103374                          BHIS    1$              ;BRANCH IF R3 LE THAN LAST ADDR
005070  000207                          RTS     PC              ;RETURN TO MAIN PROGRAM

                                .SBTTL  SUBROUTINE TO LOG AND REPORT TIMEOUTS OF MAP REGISTERS
                                        ;;*************************************************************************
                                        ;*
                                        ;*      THIS SUBROUTINE IS USED TO LOG AND REPORT THE FACT THAT A
                                        ;*      REFERENCE TO A MAPPING REGISTER TIMED OUT ON THE UNIBUS.  IT
                                        ;*      KEEPS A "LOGICAL AND" AND A "LOGICAL OR" OF EACH ADDRESS THAT
                                        ;*      TIMES OUT.
                                        ;*
                                        ;;*************************************************************************
005072                          TIMEOUT:                        ;STARTING ADDRESS OF SUBROUTINE
005072  005227                          INC     (PC)+           ;INCREMENT ONE TIME GATE
005074  177777                  TOFLAG: .WORD   -1              ;ONE TIME ENTANCE FLAG
005076  001401                          BEQ     10$             ;BRANCH IF FLAG IS NOW ZERO
005100  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 .
005102  012637  001304          10$:    MOV     (KSP)+,OLDPC    ;SAVE RETURN ADDRESS
005106  012637  001306                  MOV     (KSP)+,OLDPS    ;SAVE OLD PSW
005112  013737  177766  001266          MOV     CPUERR,PCPUER   ;SAVE CPU ERROR REGISTER
005120  013737  001266  177766          MOV     PCPUER,CPUERR   ;CLEAR CPU ERROR REGISTER
005126  023737  001266  001264          CMP     PCPUER,CPUEXP   ;SEE IF EXPECTEED CONDITION CAME UP.
005134  001402                          BEQ     1$              ;BRANCH IF IT WAS A TIMEOUT
005136  104001                          ERROR   1               ;NOT RIGHT CONDITION
005140  000414                          BR      3$              ;BRANCH TO EXIT
005142  050037  001226          1$:     BIS     R0,ADDROR       ;PERFORM LOGIAL OR OF FAILING ADDRESS
005146  005100                          COM     R0              ;GET R0 READY FOR AND
005150  040037  001224                  BIC     R0,ADRAND               ;PERFORM LOGICAL AND
005154  005100                          COM     R0              ;PUT R0 BACK AS IT WAS
005156  005737  001254                  TST     ERRCNT          ;IS THIS THE FIRST ERROR
005162  001002                          BNE     2$              ;BRANCH IF NOT FIRST ERROR
005164  104201                          ERROR   201             ;NO REGISTER RESPONSE.
005166  000401                          BR      3$              ;BRANCH TO EXIT
005170  104301                  2$:     ERROR   301             ;CONTINUE NO RESPONSE TABLE
005172  012737  177777  005074  3$:     MOV     #-1,TOFLAG      ;RESET ONE TIME GATE
005200  013746  001306                  MOV     OLDPS,-(KSP)    ;RESTORE OLD PSW
005204  013746  001304                  MOV     OLDPC,-(KSP)    ;PUSH RETURN ADDRESS BACK ON THE STACK
005210  000006                          RTT                     ;RETURN TO THE TEST

                                .SBTTL  SUBROUTINE TO REPORT MAP REGISTERS THAT WILL NOT HOLD ZERO
                                        ;;*************************************************************************
                                        ;*
                                        ;*      THIS SUBROUTINE IS CALLED EVERY TIME A MAP REGISTER IS CLEARED
                                        ;*      AND FOUND TO NOT BE ZERO.  IT KEEPS A LOGICAL "AND" AND "OR" OF
                                        ;*      THE FAILING REGISTER'S ADDRESS AND DATA AND REPORTS ALL ERRORS.
                                        ;*      AT THE END OF THE TEST A SUMMARY ERROR MESSAGE IS GIVEN WHICH
                                        ;*      WILL REPORT THE LOGICAL "AND" AND "OR" OF THE ADDRESSES AND DATA.
```

```
                                        ;*
                                        ;;*************************************************************************
005212                                  WRITEERROR:
005212  012637  001304                          MOV     (KSP)+,OLDPC    ;SAVE RETURN ADDRESS IN CASE OF ERROR LOOP
005216  050037  001226                          BIS     R0,ADDROR       ;LOGICAL "OR" OF BAD ADDRESS
005222  005100                                  COM     R0              ;GET R0 READY FOR AND
005224  040037  001224                          BIC     R0,ADRAND               ;PERFORM LOGICAL AND
005230  005100                                  COM     R0              ;PUT R0 BACK AS IT WAS
005232  053737  001170  001232                  BIS     $TMP0,DATAOR    ;LOGICAL "OR" OF BAD DATA
005240  005137  001170                          COM     $TMP0           ;GET $TMP0 READY FOR AND
005244  043737  001170  001230                  BIC     $TMP0,DATAND            ;PERFORM LOGICAL AND
005252  005137  001170                          COM     $TMP0           ;PUT $TMP0 BACK AS IT WAS
005256  005737  001254                          TST     ERRCNT          ;SEE IF THIS IS FIRST ERROR
005262  001002                                  BNE     1$              ;BRANCH IF NOT FIRST ERROR
005264  104202                                  ERROR   202             ;ERROR TYPE OUT ITEM 4
005266  000401                                  BR      2$              ;SKIP NEXT STATEMENT
005270  104302                          1$:     ERROR   302             ;ERROR TYPE OUT ITEM 5
005272  000177  174006                  2$:     JMP     @OLDPC          ;RETURN TO THE TEST

                                        .SBTTL  SUBROUTINE TO REPORT DUAL ADDRESSING WHEN LOADING A MAP REGISTER
                                        ;;*************************************************************************
                                        ;*
                                        ;*      THIS SUBROUTINE WILL LOG AND REPORT ALL DUAL ADDRESSING ERRORS
                                        ;*      FOUND IN THE MAPPING REGISTERS.  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'.
                                        ;*
                                        ;;*************************************************************************
005276                                  DUALADR:                        ;STARTING LOCATION OF SUBROUTINE
005276  012637  001304                          MOV     (KSP)+,OLDPC    ;SAVE RETURN ADDRESS IN CASE OF LOOP
005302  050037  001226                          BIS     R0,ADDROR       ;LOGICAL OR OF WRITTEN ADDRESS
005306  005100                                  COM     R0              ;GET R0 READY FOR AND
005310  040037  001224                          BIC     R0,ADRAND               ;PERFORM LOGICAL AND
005314  005100                                  COM     R0              ;PUT R0 BACK AS IT WAS
005316  050137  001232                          BIS     R1,DATAOR       ;LOGICAL OR OF DUALED ADDRESS
005322  005101                                  COM     R1              ;GET R1 READY FOR AND
005324  040137  001230                          BIC     R1,DATAND               ;PERFORM LOGICAL AND
005330  005101                                  COM     R1              ;PUT R1 BACK AS IT WAS
005332  005737  001254                          TST     ERRCNT          ;SEE IF THIS IS FIRST ERROR
005336  001002                                  BNE     1$              ;BRANCH IF NOT FIRST ERROR
005340  104203                                  ERROR   203
005342  000401                                  BR      2$              ;BRANCH TO EXIT
005344  104303                          1$:     ERROR   303
005346  000177  173732                  2$:     JMP     @OLDPC          ;RETURN TO TEST

                                        .SBTTL  SUBROUTINE TO REPORT COUNT PATTERN ERRORS IN MAP REGISTERS
                                        ;;*************************************************************************
                                        ;*
                                        ;*      THIS SUBROUTINE IS CALLED WHENEVER AN ERROR IS DISCOVERED IN
                                        ;*      THE COUNT PATTERN IN THE MAP REGISTERS.  IT KEEPS THE LOGICAL
                                        ;*      "AND" AND "OR" OF THE FAILING REGISTER'S ADDRESS, DATA RECEIVED,
                                        ;*      AND PATTERN LOADED.  EVERY ERROR IS REPORTED AND AT THE END OF
                                        ;*      A TEST AN ERROR SUMJMARY IS GIVEN.
                                        ;*
                                        ;;*************************************************************************
```

```
2218  005352  012637  001304          COUNT:  MOV     (KSP)+,OLDPC    ;SAVE RETURN ADDRESS IN CASE OF ERROR LOOP
2219  005356  050037  001226                  BIS     R0,ADDROR       ;LOGICAL OR OF BAD ADDRESSES
2220  005362  005100                          COM     R0              ;GET R0 READY FOR AND
2221  005364  040037  001224                  BIC     R0,ADRAND               ;PERFORM LOGICAL AND
2222  005370  005100                          COM     R0              ;PUT R0 BACK AS IT WAS
2223  005372  050337  001232                  BIS     R3,DATAOR       ;LOGICAL OR OF BAD DATA
2224  005376  005103                          COM     R3              ;GET R3 READY FOR AND
2225  005400  040337  001230                  BIC     R3,DATAND               ;PERFORM LOGICAL AND
2226  005404  005103                          COM     R3              ;PUT R3 BACK AS IT WAS
2227  005406  050437  001236                  BIS     R4,PATTOR       ;LOGICAL OR OF PATTERN LOADED
2228  005412  005104                          COM     R4              ;GET R4 READY FOR AND
2229  005414  040437  001234                  BIC     R4,PATAND               ;PERFORM LOGICAL AND
2230  005420  005104                          COM     R4              ;PUT R4 BACK AS IT WAS
2231  005422  005737  001254                  TST     ERRCNT          ;IS THIS FIRST ERROR
2232  005426  001002                          BNE     1$              ;BRANCH IF NOT FIRST ERROR
2233  005430  104204                          ERROR   204             ;REPORT FIRST COUNT ERROR
2234  005432  000401                          BR      2$              ;SKIP NEXT STATEMENT
2235  005434  104304                  1$:     ERROR   304             ;REPORT MORE DATA
2236  005436  000177  173642          2$:     JMP     @OLDPC          ;RETURN TO THE TEST
2237
2238
2239                                  .SBTTL  SUBROUTINE TO REPORT COUNT PATTERN ERRORS ON UNIBUS DATA PATH
2240                                  ;;*****************************************************************
2241                                  ;*
2242                                  ;*      THIS SUBROUTINE IS CALLED WHENEVER AN ERROR IS DISCOVERED IN THE
2243                                  ;*      COUNT PATTERN THAT IS RUN OVER THE UNIBUS INTO MAIN MEMORY.  IT
2244                                  ;*      KEEPS THE LOGICAL "AND" AND "OR" OF THE PATTERN LOADED AND THE
2245                                  ;*      DATA RECEIVED, AND REPORTS ALL ERRORS.  AT THE END OF THE TEST
2246                                  ;*      IT GIVES A SUMMARY MESSAGE OF ALL THE ERRORS.
2247                                  ;*
2248                                  ;;*****************************************************************
2249  005442  012637  001304          UBCOUNT:MOV     (KSP)+,OLDPC    ;SAVE RETURN ADDRESS IN CASE OF ERROR LOOP
2250  005446  050137  001236                  BIS     R1,PATTOR       ;LOGICAL OR OF COUNT LOADED
2251  005452  005101                          COM     R1              ;GET R1 READY FOR AND
2252  005454  040137  001234                  BIC     R1,PATAND               ;PERFORM LOGICAL AND
2253  005460  005101                          COM     R1              ;PUT R1 BACK AS IT WAS
2254  005462  050037  001232                  BIS     R0,DATAOR       ;LOGICAL OR OF DATA READ
2255  005466  005100                          COM     R0              ;GET R0 READY FOR AND
2256  005470  040037  001230                  BIC     R0,DATAND               ;PERFORM LOGICAL AND
2257  005474  005100                          COM     R0              ;PUT R0 BACK AS IT WAS
2258  005476  005737  001254                  TST     ERRCNT          ;IS THIS THE FIRST ERROR?
2259  005502  001002                          BNE     1$              ;BRANCH IF NOT FIRST
2260  005504  104205                          ERROR   205             ;REPORT FIRST ERROR ON UNIBUS DATA PATH
2261  005506  000401                          BR      2$              ;SKIP NEXT INSTRUCTION
2262  005510  104305                  1$:     ERROR   305             ;REPORT MORE DATA FROM UNIBUS
2263  005512  000177  173566          2$:     JMP     @OLDPC          ;RETURN TO THE TEST
2264
2265
2266                                  .SBTTL  SUBROUTINE TO REPORT COUNT PATTERN ERRORS IN CACHE REGISTERS
2267                                  ;;*****************************************************************
2268                                  ;*
2269                                  ;*      THIS SUBROUTINE IS USED TO LOG AND REPORT THE COUNT PATTERN
2270                                  ;*      ERRORS OCCURRING WHEN TESTING THE CACHE REGISTERS.
2271                                  ;*      THE "LOGICAL OR" AND "LOGICAL AND" OF VARIOUS DATA WILL BE
2272                                  ;*      MAINTAINED AS FOLLOWS:
2273                                  ;*      DATA FETCHED FROM REGISTERS IN 'DATAOR' AND 'DATAND'
```

```
                                        ;*      PATTERN LOADED INTO THE REGISTERS IN 'PATTOR' AND 'PATAND'.
                                        ;*
                                        ;;*************************************************************************
005516                                  CACOUNT:                        ;STARTING ADDRESS OF THIS SUBROUTINE
005516  012637  001304                          MOV     (KSP)+,OLDPC    ;SAVE RETURN ADDRESS IN CASE OF LOOP
005522  050237  001236                          BIS     R2,PATTOR       ;LOGICAL OR OF PATTERN LOADED
005526  005102                                  COM     R2              ;GET R2 READY FOR AND
005530  040237  001234                          BIC     R2,PATAND               ;PERFORM LOGICAL AND
005534  005102                                  COM     R2              ;PUT R2 BACK AS IT WAS
005536  050337  001232                          BIS     R3,DATAOR       ;LOGICAL OR OF DATA FETCHED
005542  005103                                  COM     R3              ;GET R3 READY FOR AND
005544  040337  001230                          BIC     R3,DATAND               ;PERFORM LOGICAL AND
005550  005103                                  COM     R3              ;PUT R3 BACK AS IT WAS
005552  105737  001103                          TSTB    $ERFLG          ;SEE IF THIS IS THE FIRST ERROR
005556  001002                                  BNE     1$              ;BRANCH IF NOT FIRST ERROR
005560  104207                                  ERROR   207
005562  000401                                  BR      2$              ;BRANCH TO EXIT
005564  104307                          1$:     ERROR   307
005566  000177  173512                  2$:     JMP     @OLDPC          ;RETURN TO TEST

                                        .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.
                                        ;*      IF THE WORD 'CPUEXP' IS ZERO, NO TRAP WAS EXPECTED AND AN
                                        ;*      UNEXPECTED ERROR MESSAGE IS GIVEN.  IF THE WORD 'CPUEXP' IS
                                        ;*      NOT ZERO THEN THE CPU ERROR REGISTER 'CPUERR' IS COMPARED WITH
                                        ;*      'CPUEXP' TO SEE IF THE PROPER CONDITION OCCURRED.  '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
                                        ;*
                                        ;;*************************************************************************
005572  005227                          CPUER:  INC     (PC)+           ;MAKE FLAG ZERO IF FIRST TIME
005574  177777                          CPFLAG: .WORD   -1              ;NEGATIVE ONE FOR A FLAG
005576  001401                                  BEQ     10$             ;BRANCH IF FIRST TIME IN
005600  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 .
005602  012637  001304                  10$:    MOV     (KSP)+,OLDPC    ;SAVE RETURN ADDRESS IN CASE OF LOOP
005606  012637  001306                          MOV     (KSP)+,OLDPS    ;SAVE OLD PSW IN CASE OF LOOP
005612  012706  001100                          MOV     #KERSTK,KSP     ;MAKE SURE THAT THE KERNEL STACK POINTER
                                                                        ;IS AT 1100.
005616  013737  177766  001266                  MOV     CPUERR,PCPUER   ;SAVE CPU ERROR REGISTER
005624  013737  001304  001302                  MOV     OLDPC,BADPC     ;SAVE PC+2 AT TIME OF ABORT
005632  005737  001264                          TST     CPUEXP          ;SEE IF ANY CONDITION WAS EXPECTED
005636  001406                                  BEQ     2$              ;BRANCH IF NO TRAP WAS EXPECTED
005640  023737  001266  001264                  CMP     PCPUER,CPUEXP   ;SEE IF EXPECTED ERROR OCCURED
005646  001403                                  BEQ     1$              ;BRANCH IF ERROR CODES MATCH
005650  104001                                  ERROR   1               ;WRONG CPU TRAP CONDITION
```

```
2330  005652  000401                          BR      1$              ;SKIP NEXT INSTRUCTION
2331  005654  104002                  2$:     ERROR   2               ;NO CPU TRAP EXPECTED
2332  005656  013737  001266  177766  1$:     MOV     PCPUER,CPUERR   ;CLEAR CPU ERROR REGISTER
2333  005664  012737  177777  005574          MOV     #-1,CPFLAG      ;MAKE FLAG NEGATIVE ONE FOR NEXT TIME
2334  005672  013746  001306                  MOV     OLDPS,-(KSP)    ;PUSH OLD PSW BACK ON STACK
2335  005676  013746  001304                  MOV     OLDPC,-(KSP)    ;PUSH RETURN ADDRESS BACK ON STACK
2336  005702  000006                          RTT                     ;RETURN FROM INTERRUPT OR ABORT
2337
2338
2339
2340                                  .SBTTL  CACHE TRAPS AND ABORTS HANDLER ROUTINE
2341                                  ;;**************************************************************************
2342                                  ;;*
2343                                  ;;*      THIS SUBROUTINE WILL HANDLE ALL UNEXPECTED PARITY ERRORS.  IF
2344                                  ;;*      THE PARITY ERROR IS AN ABORT THE TEST THAT WAS RUNNING WILL
2345                                  ;;*      BE RESTARTED ONCE AFTER THE ABORT CONDITION IS REPORTED.  ON THE
2346                                  ;;*      SECOND ABORT IN A SINGLE TEST THE NEXT TEST IS ATTEMPTED
2347                                  ;;*      AFTER THE ABORT IS REPORTED.
2348                                  ;;*      IF THE PARITY ERROR IS A TRAP THE CACHE WILL BE CLEANED UP BY
2349                                  ;;*      REMOVING THE BAD WORD THAT MAY BE IN THE CACHE, AND THE TEST
2350                                  ;;*      WILL BE CONTINUED AFTER THE PARITY ERROR IS REPORTED.
2351                                  ;;*
2352                                  ;;**************************************************************************
2353  005704  005227                  MEMER:  INC     (PC)+           ;MAKE FLAG ZERO IF FIRST TIME
2354  005706  177777                  PAFLAG: .WORD   -1              ;NEGATIVE ONE FOR A FLAG
2355  005710  001401                          BEQ     10$             ;BRANCH IF FIRST TIME IN
2356  005712  000000                          HALT                    ;I HAVE ENTERED THIS ROUTINE BEFORE
2357                                                                  ;I FINISHED REPORTING THE FIRST ERROR
2358                                                                  ;THE SECOND ENTRY ADDRESS IS ON THE
2359                                                                  ;STACK AND THE FIRST ERROR CONDITION
2360                                                                  ;IS PROBABLY STILL LOCKED UP .
2361  005714  012737  006126  000114  10$:     MOV     #5$,CACHVEC     ;SET CACHE VECTOR TO RTI UNTIL THE
2362                                                                  ;THE CACHE HAS BEEN FIXED.
2363  005722  012637  001304                  MOV     (KSP)+,OLDPC    ;SAVE RETURN ADDRESS IN CASE OF LOOP
2364  005726  012637  001306                  MOV     (KSP)+,OLDPS    ;SAVE OLD PSW IN CASE OF LOOP
2365  005732  013737  177740  001270          MOV     @#LOADRS,PLOADR ;SAVE LOWER CACHE ADDR REG
2366  005740  013737  177742  001272          MOV     @#HIADRS,PHIADR ;SAVE HIGH BITS OF FAILING ADDR
2367  005746  013737  177744  001274          MOV     @#MEMERR,PPARER ;SAVE MEMORY ERROR REGISTER
2368  005754  013737  177746  001276          MOV     @#CONTRL,PCONTR ;SAVE CONTROL REGISTER FOR TYPE OUT
2369  005762  013737  177750  001300          MOV     @#MAINT,PMAINT  ;SAVE MAINTENENCE REGISTER
2370  005770  013737  001304  001302          MOV     OLDPC,BADPC     ;SAVE PC+2 AT TIME OF ABORT
2371  005776  032737  000014  001274          BIT     #14,PPARER      ;WAS THIS A MAIN MEMORY REFERENCE
2372  006004  001005                          BNE     1$              ;BRANCH IF MAIN MEMORY PARITY ERROR
2373  006006  012737  005704  000114          MOV     #MEMER,CACHVEC  ;LOAD ADDRESS OF THIS ROUTINE IN VECTOR
2374  006014  104003                          ERROR   3               ;UNEXPECTED PARITY ABORT
2375  006016  000415                          BR      2$              ;BRANCH TO SUBROUTINE EXIT
2376  006020  010046                  1$:     MOV     R0,-(KSP)       ;SAVE R0 ON STACK
2377  006022  013700  001270                  MOV     PLOADR,R0       ;PUT LOW 16 BITS OF BAD ADDR IN R0
2378  006026  042700  176000                  BIC     #176000,R0      ;CLEAR UPPER 6 BITS OF ADDRESS
2379  006032  074027  000002                  XOR     R0,#BIT1        ;REFERENCE OTHER WORD OF PAIR
2380  006036  005710                          TST     (R0)            ;READ AT SAME INDEX AS BAD WORD
2381  006040  012600                          MOV     (KSP)+,R0       ;RESTORE R0 FROM STACK
2382  006042  012737  005704  000114          MOV     #MEMER,CACHVEC  ;LOAD ADDRESS OF THIS ROUTINE IN VECTOR
2383  006050  104004                          ERROR   4               ;MAIN MEMORY PARITY ERROR
2384  006052  005737  001322          2$:     TST     RETRY           ;ARE YOU RETRYING THIS TEST?
2385  006056  001006                          BNE     3$              ;BRANCH IF THIS IS SECOND TRY
```

```
2386  006060  005237  001322                    INC     RETRY           ;SET RETRY FLAG
2387  006064  013737  001106  001304            MOV     $LPADR,OLDPC    ;RETURN TO START OF THIS TEST
2388  006072  000403                            BR      4$              ;BRANCH TO EXIT
2389  006074  013737  001324  001304  3$:       MOV     NXTTST,OLDPC    ;RETURN TO START OF NEXT TEST
2390                                                                    ;SINCE THIS IS THE SECOND ABORT
2391  006102  013737  001274  177744  4$:       MOV     PPARER,MEMERR   ;CLEAR MEMORY ERROR REGISTER
2392  006110  012737  177777  005706            MOV     #-1,PAFLAG      ;RESTORE A NEGATIVE ONE FOR NEXT TIME
2393  006116  013746  001306                    MOV     OLDPS,-(KSP)    ;PUSH OLD PSW BACK ON STACK
2394  006122  013746  001304                    MOV     OLDPC,-(KSP)    ;PUSH RETURN ADDRESS BACK ON STACK
2395  006126  000006                  5$:       RTT                     ;RETURN FROM INTERRUPT.
2396
2397
2398
2399                                            .SBTTL  MEMORY MANAGEMENT TRAPS AND ABORTS HANDLER ROUTINE
2400                                            ;;**************************************************************************
2401                                            ;*
2402                                            ;*      THIS  ROUTINE WILL HANDLE ALL SPURIOUS MEMORY MANAGEMENT TRAPS
2403                                            ;*      AND ABORTS.  IT WILL REPORT THE CONDITION OF ALL THE MEMORY
2404                                            ;*      MAMAGEMENT STATUS REGISTERS, AND THEN RETURN TO THE TEST AND
2405                                            ;*      TRY TO CONTINUE RUNNING.
2406                                            ;*
2407                                            ;;**************************************************************************
2408  006130  005227                  MMTRAP:   INC     (PC)+           ;MAKE FLAG ZERO IF FIRST TIME
2409  006132  177777                  MMFLAG:   .WORD   -1              ;FLAG SHOULD BE NEG ONE
2410  006134  001401                            BEQ     10$             ;BRANCH IF FIRST TIME INTO ROUTINE
2411  006136  000000                            HALT                    ;I HAVE ENTERED THIS ROUTINE BEFORE
2412                                                                    ;I FINISHED REPORTING THE FIRST ERROR
2413                                                                    ;THE SECOND ENTRY ADDRESS IS ON THE
2414                                                                    ;STACK AND THE FIRST ERROR CONDITION
2415                                                                    ;IS PROBABLY STILL LOCKED UP .
2416  006140  011637  001302          10$:      MOV     (KSP),BADPC     ;SAVE PC AT TIME OF ABORT OR TRAP
2417  006144  012637  001304                    MOV     (KSP)+,OLDPC    ;SAVE RETURN ADDRESS IN CASE OF LOOP
2418  006150  012637  001306                    MOV     (KSP)+,OLDPS    ;SAVE OLD PSW IN CASE OF LOOP
2419  006154  013737  177572  001312            MOV     MMR0,PMMR0      ;SAVE STATUS REGISTER
2420  006162  013737  177574  001314            MOV     MMR1,PMMR1      ;SAVE AUTO INC/DEC REGISTER
2421  006170  013737  177576  001316            MOV     MMR2,PMMR2      ;SAVE VIRTUAL ADDRESS REGISTER
2422  006176  104005                            ERROR   5               ;UNEXPECTED M.M. ABORT OR TRAP
2423  006200  042737  177776  177572  1$:       BIC     #177776,@#MMR0  ;CLEAR ALL BITS EXCEPT 0
2424  006206  012737  177777  006132            MOV     #-1,MMFLAG      ;RESTORE A NEGATIVE ONE TO FLAG
2425  006214  013746  001306                    MOV     OLDPS,-(KSP)    ;PUSH OLD PSW ONTO STACK
2426  006220  013746  001304                    MOV     OLDPC,-(KSP)    ;PUSH RETURN ADDRESS ON STACK
2427  006224  000006                            RTT                     ;RETURN TO MAIN PROGRAM
2428
2429
2430
2431
2432                                            .SBTTL  ************************************************************
2433                                            .SBTTL  ********************* START OF TEST CODE *********************
2434          010000                            .=10000         ;START TEST CODE AT ADDRESS 10000 (2K)
2435
2436  010000                           START:
2437  010000  012737  000340  177776            MOV     #340,@#PS       ;;LOCK OUT ALL INTERRUPTS
2438  010006  012706  001100                    MOV     #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
2439  010012  005026                            CLR     (R6)+           ;;CLEAR MEMORY LOCATION
2440  010014  022706  001136                    CMP     #$TKS,R6        ;;DONE?
2441  010020  001374                            BNE     .-6             ;;LOOP BACK IF NO
```

```
                        ********** START OF TEST CODE **********

2442  010022  012706  001100                MOV     #STACK,SP       ;;SETUP THE STACK POINTER
2443  010026  012737  002140  000020        MOV     #$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
2444  010034  012737  000340  000022        MOV     #340,@#IOTVEC+2 ;;LEVEL 7
2445  010042  012737  002460  000030        MOV     #$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
2446  010050  012737  000340  000032        MOV     #340,@#EMTVEC+2 ;;LEVEL 7
2447  010056  012737  004426  000034        MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
2448  010064  012737  000340  000036        MOV     #340,@#TRAPVEC+2;LEVEL 7
2449  010072  012737  004470  000024        MOV     #$PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
2450  010100  012737  000340  000026        MOV     #340,@#PWRVEC+2 ;;LEVEL 7
2451  010106  013737  022726  022720        MOV     $ENDCT,$EOPCT   ;;SETUP END-OF-PROGRAM COUNTER
2452  010114  005037  001204                CLR     $TIMES          ;;INITIALIZE NUMBER OF ITERATIONS
2453  010120  005037  001206                CLR     $ESCAPE         ;;CLEAR THE ESCAPE ON ERROR ADDRESS
2454  010124  112737  000001  001115        MOVB    #1,$ERMAX       ;;ALLOW ONE ERROR PER TEST
2455  010132  012737  023146  000014        MOV     #$RTRN,@#TBITVEC ;;SET "T" BIT VECTOR TO $RTRN
2456  010140  012737  000340  000016        MOV     #340,@#TBITVEC+2 ;;LEVEL 7
2457  010146  012737  000002  023146        MOV     #RTI,$RTRN      ;;SET $RTRN TO A RTI
2458  010154  012737  010202  000010        MOV     #65$,@#RESVEC   ;;TRY TO DO A RTT
2459  010162  005046                        CLR     -(SP)           ;;DUMMY PS
2460  010164  012746  010172                MOV     #64$,-(SP)      ;;AND PC
2461  010170  000006                        RTT                     ;;TRY THE RTT
2462  010172  012737  000006  023146  64$:  MOV     #RTT,$RTRN      ;;RTT IS LEGAL--SET $RTRN TO A RTT
2463  010200  000402                        BR      66$
2464  010202  062706  000010          65$:  ADD     #10,SP          ;;RTT ILLEGAL--CLEAN OFF THE STACK
2465  010206  012737  000012  000010  66$:  MOV     #RESVEC+2,@#RESVEC ;;RESTORE TRAP CATCHER
2466  010214  005037  023154                CLR     $TBIT           ;;CLEAR "T" BIT SWITCH
2467  010220  012737  010220  001106        MOV     #..,$LPADR      ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
2468  010226  012737  010226  001110        MOV     #.,$LPERR       ;;SETUP THE ERROR LOOP ADDRESS
2469  010234  005227  177777                INC     #-1             ;;FIRST TIME?
2470  010240  001042                        BNE     67$             ;;BRANCH IF NO
2471  010242  022737  023100  000042        CMP     #$ENDAD,@#42    ;;ACT-11?
2472  010250  001436                        BEQ     67$             ;;BRANCH IF YES
2473  010252  104400  010260                TYPE    ,68$            ;;TYPE ASCIZ STRING
2474  010256  000433                        BR      67$             ;;GET OVER THE ASCIZ
2475                                ;;68$:  .ASCIZ  <CRLF>?MAINDEC-11-DEKBF-B  PDP-11/70 UNIBUS MAP DIAGNOSTIC?<CRLF>
2476  010346                          67$:
2477  010346  012737  005704  000114  LOOP:  MOV     #MEMER,CACHVEC  ;LOAD PARITY ERROR SERVICE
2478                                                                ;ROUTINE ADDRESS
2479  010354  012737  000340  000116        MOV     #340,CACHVEC+2  ;SET PRIORITY SEVEN
2480  010362  012737  006130  000250        MOV     #MMTRAP,MMVEC   ;LOAD MEMORY MANAGEMENT TRAP
2481                                                                ;SERVICE ROUTINE ADDRESS
2482  010370  012737  000340  000252        MOV     #340,MMVEC+2    ;SET PRIORITY SEVEN
2483  010376  012737  005572  000004        MOV     #CPUER,ERRVEC   ;LOAD CPU TRAP SERVICE ROUTINE ADDR
2484  010404  012737  000340  000006        MOV     #340,ERRVEC+2   ;SET PRIORITY SEVEN
2485  010412  012727  000044  177770        MOV     #044,#177770    ;LOAD ROM ADDRESS OF "NOP" INTO MICRO
2486                                                                ;BREAK REGISTER.  EVERY "NOP" WILL
2487                                                                ;GENERATE A "SYNC" PULSE ON PIN
2488                                                                ;#AE1 SLOT 10
2489  010420  005037  001264                CLR     CPUEXP          ;NOT EXPECTING ANY CPU ERRORS
2490  010424  005037  177572                CLR     @#MMR0          ;START IN 16 BIT MAPPING
2491  010430  005037  172516                CLR     @#MMR3          ;DISABLE MAP AND 22-BIT MAPPING
2492  010434  012700  177777                MOV     #-1,R0          ;NEGATIVE ONE USED TO INITIALIZE FLAGS
2493  010440  010037  005574                MOV     R0,CPFLAG       ;INITIALIZE FLAGS
2494  010444  010037  005074                MOV     R0,TOFLAG       ;INITIALIZE FLAGS
2495  010450  010037  005706                MOV     R0,PAFLAG       ;INITIALIZE FLAGS
2496  010454  010037  006132                MOV     R0,MMFLAG       ;INITIALIZE FLAGS
2497  010460  010037  177744                MOV     R0,@#MEMERR     ;CLEAR PARITY ERROR REGISTER
```

```
2498  010464  010037  177766                  MOV     R0,@#CPUERR     ;CLEAR CPU ERROR REGISTER
2499
2500                                  ;*THE FOLLOWING TWO TESTS ARE USED TO VERIFY THAT SOMETHING
2501                                  ;*RESPONDS WHEN THE MAP REGISTERS AND CACHE REGISTERS ARE
2502                                  ;*REFERENCED.  THE SIGNALS THAT SHOULD BE GENERATED ARE:
2503                                  ;*'MAPB REG OP H' AND 'MAPB CACHE REG H'. UNDER PROPER
2504                                  ;*OPERATION EITHER SIGNAL SHOULD CAUSE 'BUS SSYN L' TO BE
2505                                  ;*ASSERTED AND NO TIMEOUT WILL OCCUR.
2506
2507                                  ;;*****************************************************************
2508                                  ;*TEST 1         MAP REGISTER RESPONSE TEST
2509                                  ;*
2510                                  ;*      THIS TEST IS USED TO ENSURE THAT ALL THE UNIBUS MAP REGISTERS
2511                                  ;*      CAN BE REFERENCED UNDER PROGRAM CONTROL, WITHOUT TIMING OUT.
2512                                  ;*      THE ADDRESSES OF ANY MAP REGISTERS THAT TIME OUT WILL BE REPORTED
2513                                  ;*      AND, AT THE END OF THE TEST, A SUMMARY OF THOSE REGISTERS WILL
2514                                  ;*      BE GIVEN.  THE ADDRESS DECODE CIRCUITRY IS ON 'MAPB' AND 'MAPC'.
2515                                  ;*
2516                                  ;;*****************************************************************
2517  010470                          TST1:
2518  010470  000004                         SCOPE
2519  010472  012737  010554  001324         MOV     #TST2,NXTTST    ;SAVE STARTING ADDRESS OF NEXT TEST
2520                                                                  ;FOR ESCAPE ON PARITY ERRORS
2521  010500  012737  005072  000004  20$:   MOV     #TIMEOUT,ERRVEC ;LOAD ERRVEC WITH ROUTINE ADDR.
2522  010506  012700  170200                 MOV     #MAPLO,R0       ;PUT FIRST MAP REG ADDR IN R0
2523  010512  012737  010520  001110         MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
2524  010520  000240                  1$:    NOP                     ;THIS IS A SYNC POINT FOR SCOPING
2525  010522  011002                         MOV     (R0),R2         ;READ MAP REGISTERS TO R2
2526  010524  062700  000002                 ADD     #2,R0           ;POINT TONEXT MAP REGISTER
2527  010530  022700  170376                 CMP     #MAPH37,R0      ;COMPARE LAST MAP REG ADDR WITH R0
2528  010534  103371                         BHIS    1$              ;CONTINUE READING IF NOT AT LAST REG.
2529  010536  012737  010500  001110         MOV     #20$,$LPERR     ;INITIALIZE LOOPING POINTER IN CASE
2530                                                                  ;OF INTERMITTENT FAULTS.
2531  010544  005737  001254                 TST     ERRCNT          ;SEE IF THERE WERE ANY ERRORS
2532  010550  001401                         BEQ     TST2            ;;NO ERRORS GO TO NEXT TEST
2533  010552  104006                         ERROR   6               ;SUMMARY OF MAP REGISTERS THAT TIMED OUT
2534
2535                                  ;;*****************************************************************
2536                                  ;*TEST 2         CACHE REGISTER RESPONSE TEST
2537
2538                                  ;*
2539                                  ;*      THIS TEST IS USED TO ENSURE THAT ALL THE CACHE REGISTERS
2540                                  ;*      CAN BE REFERENCED UNDER PROGRAM CONTROL, WITHOUT TIMING OUT.
2541                                  ;*      THE ADDRESSES OF ANY CACHE REGISTERS THAT TIME OUT WILL BE
2542                                  ;*      REPORTED AND, AT THE END OF THE TEST, A SUMMARY OF THOSE REGISTERS
2543                                  ;*      WILL BE GIVEN.
2544                                  ;*      THESE REGISTERS ARE TESTED HERE BECAUSE THE ADDRESS DECODE AND DATA
2545                                  ;*      PATH IS ON THE UNIBUS MAP BOARD (8141).  THE ADDRESS DECODE CIRCUITRY
2546                                  ;*      IS ON 'MAPB' AND GENERATES 'MAPB CACHE REG'.
2547                                  ;*
2548                                  ;;*****************************************************************
2549  010554                          TST2:
2550  010554  000004                         SCOPE
2551  010556  012737  010632  001324         MOV     #TST3,NXTTST    ;SAVE STARTING ADDRESS OF NEXT TEST
2552                                                                  ;FOR ESCAPE ON PARITY ERRORS
2553  010564  012700  177740          20$:   MOV     #LOADRS,R0      ;PUT ADDR OF FIRST CACHE REG IN R0
```

```
2554  010570  012737  010576  001110          MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER HERE
2555  010576  000240                  1$:     NOP                     ;THIS IS A SYNC POINT FOR SCOPING
2556  010600  011002                          MOV     (R0),R2         ;TRY TO READ ALL CACHE REGISTERS
2557  010602  062700  000002                  ADD     #2,R0           ;POINT TO NEXT CACHE REGISTER
2558  010606  022700  177752                  CMP     #HITMIS,R0      ;HAVE ALL REGISTERS BEEN REFERENCED?
2559  010612  103371                          BHIS    1$              ;BRANCH IF MORE TO TEST
2560  010614  012737  010564  001110          MOV     #20$,$LPERR     ;INITIALIZE LOOPING ERROR IN CASE
2561                                                                  ;OF INTERMITTENT FAULTS
2562  010622  005737  001254                  TST     ERRCNT          ;WERE THERE ANY ERRORS.
2563  010626  001401                          BEQ     TST3            ;;BRANCH IF NO ERRORS
2564  010630  104007                          ERROR   7               ;SUMMARY OF CACHE REGISTERS THAT TIMED OUT
2565
2566                          ;*THE NEXT FOUR (4) TESTS ARE USED TO CHECK THE GENERATION
2567                          ;*OF: 'MAPB WRITE HI WORD L' AND 'MAPB WRITE LOWORD L'.
2568                          ;*ALL REGISTERS ARE CLEARED AND THEN READ AND CHECKED FOR
2569                          ;*ZEROES.  THESE TESTS ALSO DETECT BITS IN THE DATA PATH
2570                          ;*TO AND FROM THE MAP REGISTERS OR IN THE MAP REGISTERS STUCK
2571                          ;*AT "1".  THE RECEIVERS ON PAGE MAPA AND THE B & C INPUTS
2572                          ;*TO THE MULTIPLEXERS ON PAGE MAPJ AND THE DRIVERS ON PAGE
2573                          ;*MAPJ MAKE UP THE DATA PATH UNDER TEST HERE.
2574
2575                          ;;*****************************************************************
2576                          ;*TEST 3        CLEAR AND READ LOW 20 REGISTERS LOWER 16 BITS
2577                          ;*
2578                          ;*      THIS TEST WILL ENSURE THAT MAPL00 - MAPL17 WILL ALL HOLD ZERO.
2579                          ;*      IF ANY REGISTERS ARE NOT ZERO AFTER BEING CLEARED, THEIR ADDRESS AND
2580                          ;*      CONTENTS ARE REPORTED.  IF ALL THE REGISTERS HAVE RANDOM DATA THEN
2581                          ;*      PROBABLY NO WRITE PULSE IS BEING GENERATED, BUT IF THE ERROR IS IN
2582                          ;*      ONLY ONE REGISTER OR IN ONE BIT IT MIGHT BE MORE DIFFICULT TO FIND.
2583                          ;*      IT IS POSSIBLE THAT THE COUNT PATTERN TEST FOR MAPL00 - MAPL17 WILL
2584                          ;*      GIVE MORE HELP.
2585                          ;*
2586                          ;;*****************************************************************
2587  010632                  TST3:
2588  010632  000004                  SCOPE
2589  010634  012737  010730  001324          MOV     #TST4,NXTTST    ;SAVE STARTING ADDRESS OF NEXT TEST
2590                                                                  ;FOR ESCAPE ON PARITY ERRORS
2591  010642  012737  005572  000004          MOV     #CPUER,ERRVEC   ;LOAD CPU TRAP SERVICE ROUTINE ADDR
2592  010650  012700  170200          20$:    MOV     #MAPL0,R0       ;LOAD STARTING MAP REGISTER ADDR
2593  010654  012737  010662  001110          MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER HERE
2594  010662  000240                  1$:     NOP                     ;THIS IS A SYNC POINT FOR SCOPING
2595  010664  005010                          CLR     (R0)            ;CLEAR MAP REGISTER
2596  010666  011037  001170                  MOV     (R0),$TMP0      ;SEE IF MAP REGISTER IS ZERO
2597  010672  001402                          BEQ     2$              ;BRANCH IF REGISTER IS ZERO
2598  010674  004737  005212                  JSR     PC,WRITEERROR   ;REPORT AND LOG ERROR
2599  010700  062700  000004          2$:     ADD     #4,R0           ;POINT TO NEXT MAP REG LOWER 16 BITS
2600  010704  022700  170274                  CMP     #MAPL17,R0      ;SEE IF THIS SECTION IS TESTED
2601  010710  103364                          BHIS    1$              ;BRANCH IF NOT
2602  010712  012737  010650  001110          MOV     #20$,$LPERR     ;INITIALIZE LOOPING POINTER IN CASE
2603                                                                  ;OF INTERMITTENT FAULTS.
2604  010720  005737  001254                  TST     ERRCNT          ;WERE THERE ANY ERRORS ON THIS TEST
2605  010724  001401                          BEQ     TST4            ;;BRANCH IF NO ERRORS
2606  010726  104010                          ERROR   10              ;SUMMARY OF MAP REGS THAT COULD NOT
2607                                                                  ;HOLD ZERO
2608
2609
```

```
                                ;;*****************************************************************
                                ;*TEST 4        CLEAR AND READ LOW 20 REGISTERS UPPER 6 BITS
                                ;*
                                ;*      THIS TEST WILL ENSURE THAT MAPH00 - MAPH17 WILL ALL HOLD ZERO.
                                ;*      IF ANY REGISTERS ARE NOT ZERO AFTER BEING CLEARED, THEIR ADDRESS AND
                                ;*      CONTENTS ARE REPORTED.  IF ALL THE REGISTERS HAVE RANDOM DATA THEN
                                ;*      PROBABLY NO WRITE PULSE IS BEING GENERATED, BUT IF THE ERROR IS IN
                                ;*      ONLY ONE REGISTER OR IN ONE BIT IT MIGHT BE MORE DIFFICULT TO FIND.
                                ;*      IT IS POSSIBLE THAT THE COUNT PATTERN TEST FOR MAPH00 - MAPH17 WILL
                                ;*      GIVE MORE HELP.
                                ;*
                                ;;*****************************************************************
010730                          TST4:
010730  000004                          SCOPE
010732  012737  011020  001324          MOV     #TST5,NXTTST    ;SAVE STARTING ADDRESS OF NEXT TEST
                                                                ;FOR ESCAPE ON PARITY ERRORS
010740  012700  170202          20$:    MOV     #MAPH0,R0       ;LOAD STARTING MAP REGISTER ADDR
010744  012737  010752  001110          MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER HERE
010752  000240                  1$:     NOP                     ;THIS IS A SYNC POINT FOR SCOPING
010754  005010                          CLR     (R0)            ;CLEAR MAP REGISTER
010756  011037  001170                  MOV     (R0),$TMP0      ;SEE IF MAP REGISTER IS ZERO
010762  001402                          BEQ     2$              ;BRANCH IF REGISTER IS ZERO
010764  004737  005212                  JSR     PC,WRITEERROR   ;REPORT AND LOG ERROR
010770  062700  000004          2$:     ADD     #4,R0           ;POINT TO NEXT MAP REG UPPER 6 BITS
010774  022700  170276                  CMP     #MAPH17,R0      ;SEE IF TESTING IS OVER
011000  103364                          BHIS    1$              ;BRANCH IF MORE REGS TO TEST
011002  012737  010740  001110          MOV     #20$,$LPERR     ;INITIALIZE LOOPING POINTER IN CASE
                                                                ;OF INTERMITTENT FAULTS.
011010  005737  001254                  TST     ERRCNT          ;WHERE THERE ANY ERRORS ON THIS
011014  001401                          BEQ     TST5            ;;BRANCH IF NO ERRORS
011016  104011                          ERROR   11              ;SUMMARY OF MAP REGS THAT COULD NOT
                                                                ;HOLD ZERO


                                ;;*****************************************************************
                                ;*TEST 5        CLEAR AND READ HIGH 20 REGISTERS LOWER 16 BITS
                                ;*
                                ;*      THIS TEST WILL ENSURE THAT MAPL20 - MAPL37 WILL ALL HOLD ZERO.
                                ;*      IF ANY REGISTERS ARE NOT ZERO AFTER BEING CLEARED, THEIR ADDRESS AND
                                ;*      CONTENTS ARE REPORTED.  IF ALL THE REGISTERS HAVE RANDOM DATA THEN
                                ;*      PROBABLY NO WRITE PULSE IS BEING GENERATED, BUT IF THE ERROR IS IN
                                ;*      ONLY ONE REGISTER OR IN ONE BIT IT MIGHT BE MORE DIFFICULT TO FIND.
                                ;*      IT IS POSSIBLE THAT THE COUNT PATTERN TEST FOR MAPL20 - MAPL37 WILL
                                ;*      GIVE MORE HELP.
                                ;*
                                ;;*****************************************************************
011020                          TST5:
011020  000004                          SCOPE
011022  012737  011110  001324          MOV     #TST6,NXTTST    ;SAVE STARTING ADDRESS OF NEXT TEST
                                                                ;FOR ESCAPE ON PARITY ERRORS
011030  012700  170300          20$:    MOV     #MAPL20,R0      ;LOAD STARTING MAP REGISTER ADDR
011034  012737  011042  001110          MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER HERE
011042  000240                  1$:     NOP                     ;THIS IS A SYNC POINT FOR SCOPING
011044  005010                          CLR     (R0)            ;CLEAR MAP REGISTER
011046  011037  001170                  MOV     (R0),$TMP0      ;SEE IF MAP REGISTER IS ZERO
011052  001402                          BEQ     2$              ;BRANCH IF REGISTER IS ZERO
```

```
2666  011054  004737  005212                  JSR     PC,WRITEERROR   ;REPORT AND LOG ERROR
2667  011060  062700  000004          2$:     ADD     #4,R0           ;POINT TO NEXT REGISTER
2668  011064  022700  170374                  CMP     #MAPL37,R0      ;SEE IF THIS SECTION IS TESTED
2669  011070  103364                          BHIS    1$              ;BRANCH IF NOT
2670  011072  012737  011030  001110          MOV     #20$,$LPERR     ;INITIALIZE LOOPING POINTER IN CASE
2671                                                                  ;OF INTERMITTENT FAULTS.
2672  011100  005737  001254                  TST     ERRCNT          ;WERE THERE ANY ERRORS ON THIS
2673  011104  001401                          BEQ     TST6            ;;BRANCH IF NO ERRORS
2674  011106  104010                          ERROR   10              ;SUMMARY OF MAP REGS THAT COULD NOT
2675                                                                  ;HOLD ZERO
2676
2677
2678                                  ;;*****************************************************************
2679                                  ;*TEST 6        CLEAR AND READ HIGH 20 REGISTERS UPPER 6 BITS
2680                                  ;*
2681                                  ;*      THIS TEST WILL ENSURE THAT MAPH20 - MAPH37 WILL ALL HOLD ZERO.
2682                                  ;*      IF ANY REGISTERS ARE NOT ZERO AFTER BEING CLEARED, THEIR ADDRESS AND
2683                                  ;*      CONTENTS ARE REPORTED.  IF ALL THE REGISTERS HAVE RANDOM DATA THEN
2684                                  ;*      PROBABLY NO WRITE PULSE IS BEING GENERATED, BUT IF THE ERROR IS IN
2685                                  ;*      ONLY ONE REGISTER OR IN ONE BIT IT MIGHT BE MORE DIFFICULT TO FIND.
2686                                  ;*      IT IS POSSIBLE THAT THE COUNT PATTERN TEST FOR MAPH20 - MAPH37 WILL
2687                                  ;*      GIVE MORE HELP.
2688                                  ;*
2689                                  ;;*****************************************************************
2690  011110                          TST6:
2691  011110  000004                          SCOPE
2692  011112  012737  011200  001324          MOV     #TST7,NXTTST    ;SAVE STARTING ADDRESS OF NEXT TEST
2693                                                                  ;FOR ESCAPE ON PARITY ERRORS
2694  011120  012700  170302          20$:    MOV     #MAPH20,R0      ;LOAD STARTING MAP REGISTER ADDR
2695  011124  012737  011132  001110          MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER HERE
2696  011132  000240                  1$:     NOP                     ;THIS IS A SYNC POINT FOR SCOPING
2697  011134  005010                          CLR     (R0)            ;CLEAR MAP REGISTER
2698  011136  011037  001170                  MOV     (R0),$TMP0      ;SEE IF MAP REGISTER IS ZERO
2699  011142  001402                          BEQ     2$              ;BRANCH IF REGISTER IS ZERO
2700  011144  004737  005212                  JSR     PC,WRITEERROR   ;REPORT AND LOG ERROR
2701  011150  062700  000004          2$:     ADD     #4,R0           ;POINT TO NEXT HIGH REGISTER
2702  011154  022700  170376                  CMP     #MAPH37,R0      ;SEE IF THERE ARE MORE REGS LEFT
2703  011160  103364                          BHIS    1$              ;BRANCH IF MORE REGS TO TEST
2704  011162  012737  011120  001110          MOV     #20$,$LPERR     ;INITIALIZE LOOPING POINTER IN CASE
2705                                                                  ;OF INTERMITTENT FAULTS.
2706  011170  005737  001254                  TST     ERRCNT          ;WERE THERE ANY ERRORS ON THIS TEST
2707  011174  001401                          BEQ     TST7            ;;BRANCH IF NO ERRORS
2708  011176  104011                          ERROR   11              ;SUMMARY OF MAP REGS THAT COULD NOT
2709                                                                  ;HOLD ZERO
2710
2711                                  ;*IN THE NEXT TWO TESTS A COUNT PATTERN IS RUN THROUGH
2712                                  ;*MAP REGISTER 00 TO TEST THE DATA PATH TO AND FROM THE
2713                                  ;*MAP REGISTERS.  THE DATA RECEIVERS ARE ON PAGE MAPA, AND
2714                                  ;*THE DATA DRIVERS ARE ON PAGE MAPJ.  THE BINPUTS TO THE
2715                                  ;*MULTIPLEXERS ON PAGE MAPJ ARE USED WHEN READING THE LOWER 16 BITS (TEST 7) AND
2716                                  ;*THE C INPUTS ARE USED WHEN READING THE UPPER 6 BITS OF THE
2717                                  ;*MAP REGISTER (TEST 10).  IF REGISTER 0 FAILS THEN REGISTER 20
2718                                  ;*IS TRIED AND THE ERROR IS REPORTED EVEN IF REGISTER 20 SUCCEEDS.
2719
2720                                  ;;*****************************************************************
2721                                  ;*TEST 7        DATA PATH TEST TO MAP REGISTER LOWER 16 BITS
```

```
                                ;*
                                ;*      THIS TEST WILL ENSURE THAT THE DATA PATH TO AND FROM THE UNIBUS
                                ;*      MAP REGISTERS IS FUNCTIONING PROPERLY.  IT RUNS A COUNT PATTERN
                                ;*      THROUGH MAPLOO, AND IF IT DETECTS AN ERROR THE EXPECTED COUNT
                                ;*      AND THE RECEIVED COUNT ARE REPORTED.  THE TEST THEN TRIES TO RUN
                                ;*      THE COUNT THROUGH MAPL20, IN CASE THE ERROR IS IN THE LOWER
                                ;*      GROUP OF REGISTERS AND NOT IN THE DATA PATH.
                                ;*      THE DATA INPUT TO THE MAP REGISTERS IS RECEIVED FROM THE UNIBUS
                                ;*      ON 'MAPA' AND THE OUTPUT TO THE UNIBUS IS DRIVEN ON 'MAPJ'.
                                ;*
                                ;**********************************************************************
011200                          TST7:
011200  000004                          SCOPE
011202  012737  011324  001324          MOV     #TST10,NXTTST   ;SAVE STARTING ADDRESS OF NEXT TEST
                                                                ;FOR ESCAPE ON PARITY ERRORS
011210  012737  000012  001204          MOV     #12,$TIMES      ;DO 12 ITERATIONS
011216  005003                  20$:    CLR     R3              ;START COUNT PATTERN AT 0
011220  005002                          CLR     R2              ;START COUNT PATTERN AT 0
011222  012737  011230  001110          MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
011230  000240                  1$:     NOP                     ;THIS IS A SYNC POINT FOR SCOPING
011232  010237  170200                  MOV     R2,MAPLOO       ;LOAD MAP REG 0 LOW 16 BITS
011236  013700  170200                  MOV     MAPLOO,R0       ;READ CONTENTS OF MAP REGISTER 0
011242  020200                          CMP     R2,R0           ;COMPARE DATA & PATTERN
011244  001402                          BEQ     2$              ;BRANCH IF DATA DOES NOT MATCH
011246  104012                          ERROR   12              ;POSSIBLE DATA PATH FAILURE
011250  000404                          BR      3$              ;BRANCH TO TEST DATA PATH WITH
                                                                ;A REGISTER OF THE NEXT GROUP
011252  062702  000002          2$:     ADD     #2,R2           ;ADD 2 TO COUNT PATTERN
011256  001364                          BNE     1$              ;BRANCH IF COUNT NOT COMPLETE.
011260  000416                          BR      10$             ;DATA PATH OKAY FOR LOW 16 BITS
011262  012737  011270  001110  3$:     MOV     #12$,$LPERR     ;SET LOOP ON ERROR POINTER TO 12$
011270  000240                  12$:    NOP                     ;THIS IS A SYNC POINT FOR SCOPING
011272  010337  170300                  MOV     R3,MAPL20       ;LOAD MAP REG 20 LOW 16 BITS
011276  013701  170300                  MOV     MAPL20,R1       ;READ DATA STORED IN MAP REGISTER 20
011302  020301                          CMP     R3,R1           ;COMPARE DATA & PATTERN
011304  001003                          BNE     4$              ;CAUGHT ERROR,NOW SEE IF SAME.
011306  062703  000002                  ADD     #2,R3           ;ADD 2 TO COUNT PATTERN
011312  001357                          BNE     2$              ;BRANCH IF COUNT NOT 0
011314  104013                  4$:     ERROR   13              ;PROBABLY IS A REAL DATA PATH ERROR
011316  012737  011216  001110  10$:    MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST


                                ;**********************************************************************
                                ;*TEST 10       DATA PATH TEST TO MAP REGISTER UPPER 6 BITS
                                ;*
                                ;*      THIS TEST WILL ENSURE THAT THE DATA PATH TO AND FROM THE UNIBUS
                                ;*      MAP REGISTERS IS FUNCTIONING PROPERLY.  IT RUNS A COUNT PATTERN
                                ;*      THROUGH MAPHOO, AND IF IT DETECTS AN ERROR THE EXPECTED COUNT
                                ;*      AND THE RECEIVED COUNT ARE REPORTED.  THE TEST THEN TRIES TO RUN
                                ;*      THE COUNT THROUGH MAPH20, IN CASE THE ERROR IS IN THE LOWER
                                ;*      GROUP OF REGISTERS AND NOT IN THE DATA PATH.
                                ;*      THE DATA INPUT TO THE MAP REGISTERS IS RECEIVED FROM THE UNIBUS
                                ;*      ON 'MAPA' AND THE OUTPUT TO THE UNIBUS IS DRIVEN ON 'MAPJ'.
                                ;*
                                ;**********************************************************************
011324                          TST10:
```

```
2778  011324  000004                          SCOPE
2779  011326  012737  011446  001324          MOV     #TST11,NXTTST   ;SAVE STARTING ADDRESS OF NEXT TEST
2780                                                                  ;FOR ESCAPE ON PARITY ERRORS
2781  011334  005002                  20$:     CLR     R2              ;START COUNT PATTERN AT 0
2782  011336  005003                          CLR     R3              ;START COUNT PATTERN AT 0
2783  011340  012737  011346  001110          MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
2784  011346  000240                  1$:     NOP                     ;THIS IS A SYNC POINT FOR SCOPING
2785  011350  010237  170202                  MOV     R2,@#MAPH0      ;LOAD MAP REG0 HIGH 6 BITS
2786  011354  013700  170202                  MOV     MAPH00,R0       ;READ BACK MAP REG 00 UPPER SIX BITS
2787  011360  020200                          CMP     R2,R0           ;COMPARE DATA AND PATTERN LOADED
2788  011362  001402                          BEQ     2$              ;BRANCH IF DATA IS CORRECT
2789  011364  104012                          ERROR   12              ;POSSIBLE DATA PATH ERROR
2790  011366  000405                          BR      3$              ;BRANCH TO TRY SECOND BANK OF MAP REGS
2791  011370  005202                  2$:     INC     R2              ;CHANGE DATA PATTERN
2792  011372  022702  000100                  CMP     #100,R2         ;HAS COUNT REACH MAXIMUM?
2793  011376  001363                          BNE     1$              ;BRANCH IF TEST NOT OVER
2794  011400  000417                          BR      10$             ;UPPER SIX BIT DATA PATH IS OKAY
2795  011402  012737  011410  001110  3$:     MOV     #12$,$LPERR     ;SET LOOP ON ERROR POINTER TO 12$
2796  011410  000240                  12$:    NOP                     ;THIS IS A SYNC POINT FOR SCOPING
2797  011412  010337  170302                  MOV     R3,@#MAPH20     ;LOAD MAP REG 20 HIGH 6 BITS
2798  011416  013701  170302                  MOV     MAPH20,R1       ;READ BACK MAP REG 20 UPPER SIX BITS
2799  011422  020301                          CMP     R3,R1           ;COMPARE DATA AND PATTERN
2800  011424  001004                          BNE     4$              ;BRANCH IF DATA DOES NOT MATCH
2801  011426  005203                          INC     R3              ;CHANGE DATA PATTERN
2802  011430  022703  000100                  CMP     #100,R3         ;HAS COUNT REACHED MAXIMUM?
2803  011434  001355                          BNE     2$              ;BRANCH IF TEST NOT OVER
2804  011436  104013                  4$:     ERROR   13              ;PROBABLY IS A REAL DATA PATH ERROR
2805  011440  012737  011334  001110  10$:    MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
2806
2807                                          ;;*****************************************************************
2808                                          ;*TEST 11       DUAL ADDRESSING ON LOADING AND READING MAP REGISTERS
2809                                          ;*
2810                                          ;*      THIS TEST ENSURES THAT ONLY ONE UNIBUS MAP REGISTER IS LOADED
2811                                          ;*      DURING A "MOV   #DATA,@#MAPREG" INSTRUCTION.  ALL MAP REGISTERS
2812                                          ;*      ARE CLEARED AND ONE REGISTER AT A TIME, STARTING WITH MAPL00,
2813                                          ;*      IS LOADED A NEGATIVE ONE.  THEN, ALL MAP REGISTERS ARE READ,
2814                                          ;*      STARTING WITH MAPH37, AND VERIFIED TO BE ZERO.  ANY REGISTER
2815                                          ;*      THAT IS NOT ZERO AND WHOSE UNIBUS ADDRESS DOES NOT MATCH THAT
2816                                          ;*      OF THE REGISTER UNDER TEST IS REPORTED TO BE IN ERROR.  AT THE
2817                                          ;*      END OF THE TEST A SUMMARY OF ALL DUALED REGISTERS IS GIVEN.
2818                                          ;*      THE SIGNALS THAT ARE TESTED HERE ARE ON 'MAPC' AND ARE:
2819                                          ;*      "EN LOREG" & "EN HIREG".  THE "A" INPUTS TO THE ADDRESS
2820                                          ;*      MULTIPLEXER SHOULD BE THE ONES IN USE DURING A READ OR WRITE
2821                                          ;*      TO THE UNIBUS MAP REGISTERS.
2822                                          ;*
2823                                          ;*****************************************************************
2824  011446                          TST11:
2825  011446  000004                          SCOPE
2826  011450  012737  011572  001324          MOV     #TST12,NXTTST   ;SAVE STARTING ADDRESS OF NEXT TEST
2827                                                                  ;FOR ESCAPE ON PARITY ERRORS
2828  011456  012737  000144  001204          MOV     #144,$TIMES     ;;DO 144 ITERATIONS
2829  011464  012700  170200          20$:    MOV     #MAPL0,R0       ;LOAD FIRST MAP REG ADDR INTO R0
2830  011470  012737  011476  001110          MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER HERE
2831  011476  012701  170376          1$:     MOV     #MAPH37,R1      ;PUT ADDRESS OF HIGHEST MAP REG IN R1
2832  011502  004737  005054                  JSR     PC,CLRMAP       ;CLEAR ALL MAP REGISTERS IN CASE OF
2833                                                                  ;A 'FLAKEY' PROBLEM
```

```
2834  011506  000240                          NOP             ;THIS IS A SYNC POINT FOR SCOPING
2835  011510  012710  177777                  MOV     #177777,(R0)    ;LOAD MAP REGISTER UNDER TEST
2836  011514  011102                  2$:     MOV     (R1),R2         ;READ MAP REGS STARTING FROM TOP
2837  011516  001404                          BEQ     3$              ;GO TO 3$ IF ZERO
2838  011520  020100                          CMP     R1,R0           ;SEE IF NON-ZERO ADDRS MATCH
2839  011522  001402                          BEQ     3$              ;GO TO 3$ IF MATCH
2840  011524  004737  005276                  JSR     PC,DUALADR      ;JUMP TO SUBROUTINE TO LOG ALL ERRORS
2841  011530  162701  000002          3$:     SUB     #2,R1           ;POINT TO NEXT MAP REGISTER
2842  011534  022701  170200                  CMP     #MAPL0,R1       ;HAVE ALL REGISTERS BEEN READ
2843  011540  101765                          BLOS    2$              ;IF NOT CONTINUE TEST
2844  011542  062700  000002                  ADD     #2,R0           ;POINT TO NEXT MAP REGISTER
2845  011546  022700  170376                  CMP     #MAPH37,R0      ;SEE IF TEST IS OVER
2846  011552  103351                          BHIS    1$              ;CONTINUE TESTING
2847  011554  012737  011464  001110          MOV     #20$,$LPERR     ;INITIALIZE LOOPING POINTER IN CASE
2848                                                                  ;OF INTERMITTENT FAULTS.
2849  011562  005737  001254                  TST     ERRCNT          ;SEE IF THERE WERE ANY ERRORS
2850  011566  001401                          BEQ     TST12           ;;GOTO NEXT TEST IF NO ERRORS
2851  011570  104014                          ERROR   14              ;SUMMARY OF DUAL ADDRESSING ERRORS
2852                                                                  ;ON LOADING MAP REGISTERS
2853
2854
2855                                  ;*THE NEXT FOUR(4) TESTS RUN A COUNT PATTERN THROUGH EACH
2856                                  ;*MAP REGISTER, CHECKING FOR STUCK BITS, AND SHORTED PINS.
2857
2858                                  ;;*****************************************************************
2859                                  ;*TEST 12       COUNT PATTERN LOW 20 MAP REGISTERS LOWER 16 BITS
2860                                  ;*
2861                                  ;*      THIS TEST WILL RUN A COUNT PATTERN THROUGH UNIBUS MAP REGISTERS
2862                                  ;*      MAPL00 - MAPL17.  IF THE COUNT RECIEVED DOES NOT MATCH THE
2863                                  ;*      EXPECTED PATTERN THEN THE MAP REGISTER ADDRESS, DATA RECEIVED,
2864                                  ;*      PATTERN LOADED, AND PATTERN EXPECTED ARE REPORTED.  AT THE END
2865                                  ;*      OF THE TEST A SUMMARY OF ALL ERRORS IS GIVEN SO THAT YOU CAN
2866                                  ;*      DETERMINE IF THE ERROR IS BIT SENSITIVE OR REGISTER SENSITIVE.
2867                                  ;*      THE REGISTERS ARE ON 'MAPC' AND THE OUTPUT MULTIPLEXER IS ON
2868                                  ;*      'MAPJ'.
2869                                  ;;*****************************************************************
2870  011572                          TST12:
2871  011572  000004                          SCOPE
2872  011574  012737  011716  001324          MOV     #TST13,NXTTST   ;SAVE STARTING ADDRESS OF NEXT TEST
2873                                                                  ;FOR ESCAPE ON PARITY ERRORS
2874  011602  012737  000012  001204          MOV     #12,$TIMES      ;;DO 12 ITERATIONS
2875  011610  005002                  20$:    CLR     R2              ;START WITH AN ALL ZERO PATTERN
2876  011612  012737  011632  001110          MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
2877  011620  010204                  1$:     MOV     R2,R4           ;PUT COUNT IN R4 FOR MASK
2878  011622  042704  000001                  BIC     #000001,R4      ;CLEAR BITS IN MASK FOR PROPER COMPARE
2879  011626  012700  170200                  MOV     #MAPL00,R0      ;LOAD STARTING MAP REGISTER ADDRESS
2880  011632  000240                  2$:     NOP                     ;THIS IS A SYNC POINT FOR SCOPING
2881  011634  010210                          MOV     R2,(R0)         ;LOAD MAP REGISTER WITH COUNT PATTERN
2882  011636  011003                          MOV     (R0),R3         ;READ MAP REGISTER INTO R3
2883  011640  020403                          CMP     R4,R3           ;COMPARE MASK WITH DATA
2884  011642  001402                          BEQ     3$              ;BRANCH IF DATA MATCHES
2885  011644  004737  005352                  JSR     PC,COUNT        ;JUMP TO COUNT TO LOG ALL ERRORS
2886  011650  005003                  3$:     CLR     R3              ;CLEAR DATA HOLDER
2887  011652  062700  000004                  ADD     #4,R0           ;POINT TO NEXT MAP REGISTER UNDER TEST
2888  011656  022700  170274                  CMP     #MAPL17,R0      ;SEE IF ALL REGISTERS HAVE BEEN LOADED
2889  011662  103363                          BHIS    2$              ;BRANCH IF STILL MORE TO GO
```

```
                                                CMP     #177777,R2      ;SEE IF COUNT HAS CYCLED
                                                BEQ     4$              ;BRANCH IF TEST IS DONE
                                                ADD     #401,R2         ;CHANGE COUNT PATTERN
                                                BR      1$              ;CONTINUE TESTING
                                        4$:     MOV     #20$,$LPERR     ;INITIALIZE LOOPING POINTER IN CASE
                                                                        ;OF INTERMITTENT FAULTS.
                                                TST     ERRCNT          ;SEE IF THERE WERE ANY ERRORS
                                                BEQ     TST13           ;;BRANCH IF NO ERRORS
                                                ERROR   15              ;SUMMARY OF COUNT PATTERN FAILURES
                                                                        ;IN MAPPING REGISTERS


                                ;*****************************************************************
                                ;*TEST 13       COUNT PATTERN LOW 20 MAP REGISTERS UPPER 6 BITS
                                ;*
                                ;*      THIS TEST WILL RUN A COUNT PATTERN THROUGH UNIBUS MAP REGISTERS
                                ;*      MAPH00 - MAPH17.  IF THE COUNT RECIEVED DOES NOT MATCH THE
                                ;*      EXPECTED PATTERN THEN THE MAP REGISTER ADDRESS, DATA RECEIVED,
                                ;*      PATTERN LOADED, AND PATTERN EXPECTED ARE REPORTED.  AT THE END
                                ;*      OF THE TEST A SUMMARY OF ALL ERRORS IS GIVEN SO THAT YOU CAN
                                ;*      DETERMINE IF THE ERROR IS BIT SENSITIVE OR REGISTER SENSITIVE.
                                ;*      THE REGISTERS ARE ON 'MAPC' AND THE OUTPUT MULTIPLEXER IS ON
                                ;*      'MAPJ'.
                                ;*****************************************************************
                                TST13:
                                                SCOPE
                                                MOV     #TST14,NXTTST   ;SAVE STARTING ADDRESS OF NEXT TEST
                                                                        ;FOR ESCAPE ON PARITY ERRORS
                                                MOV     #144,$TIMES     ;;DO 144 ITERATIONS
                                        20$:    CLR     R2              ;START WITH AN ALL ZERO PATTERN
                                                MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
                                        1$:     MOV     R2,R4           ;PUT COUNT IN R4 FOR MASK
                                                BIC     #177700,R4      ;CLEAR BITS IN MASK FOR PROPER COMPARE
                                                MOV     #MAPH00,R0      ;LOAD STARTING MAP REGISTER ADDRESS
                                        2$:     NOP                     ;THIS IS A SYNC POINT FOR SCOPING
                                                MOV     R2,(R0)         ;LOAD MAP REGISTER WITH COUNT PATTERN
                                                MOV     (R0),R3         ;READ MAP REGISTER INTO R3
                                                CMP     R4,R3           ;COMPARE MASK WITH DATA
                                                BEQ     3$              ;BRANCH IF DATA MATCHES
                                                JSR     PC,COUNT        ;JUMP TO COUNT TO LOG ALL ERRORS
                                        3$:     CLR     R3              ;CLEAR DATA HOLDER
                                                ADD     #4,R0           ;POINT TO NEXT MAP REGISTER UNDER TEST
                                                CMP     #MAPH17,R0      ;SEE IF ALL REGISTERS HAVE BEEN LOADED
                                                BHIS    2$              ;BRANCH IF STILL MORE TO GO
                                                CMP     #177,R2 ;SEE IF COUNT HAS CYCLED
                                                BEQ     4$              ;BRANCH IF TEST IS DONE
                                                ADD     #1,R2           ;CHANGE COUNT PATTERN
                                                BR      1$              ;CONTINUE TESTING
                                        4$:     MOV     #20$,$LPERR     ;INITIALIZE LOOPING POINTER IN CASE
                                                                        ;OF INTERMITTENT FAULTS.
                                                TST     ERRCNT          ;SEE IF THERE WERE ANY ERRORS
                                                BEQ     TST14           ;;BRANCH IF NO ERRORS
                                                ERROR   16              ;SUMMARY OF COUNT PATTERN FAILURES
                                                                        ;IN MAPPING REGISTERS
```

```
                                ;;*****************************************************************
                                ;*TEST 14       COUNT PATTERN HIGH 20 MAP REGISTERS LOWER 16 BITS
                                ;*
                                ;*      THIS TEST WILL RUN A COUNT PATTERN THROUGH UNIBUS MAP REGISTERS
                                ;*      MAPL20 - MAPL37.  IF THE COUNT RECIEVED DOES NOT MATCH THE
                                ;*      EXPECTED PATTERN THEN THE MAP REGISTER ADDRESS, DATA RECEIVED,
                                ;*      PATTERN LOADED, AND PATTERN EXPECTED ARE REPORTED.  AT THE END
                                ;*      OF THE TEST A SUMMARY OF ALL ERRORS IS GIVEN SO THAT YOU CAN
                                ;*      DETERMINE IF THE ERROR IS BIT SENSITIVE OR REGISTER SENSITIVE.
                                ;*      THE REGISTERS ARE ON 'MAPD' AND THE OUTPUT MULTIPLEXER IS ON
                                ;*      'MAPJ'.
                                ;;*****************************************************************
012042                          TST14:
012042  000004                          SCOPE
012044  012737  012166  001324          MOV     #TST15,NXTTST   ;SAVE STARTING ADDRESS OF NEXT TEST
                                                                ;FOR ESCAPE ON PARITY ERRORS
012052  012737  000012  001204          MOV     #12,$TIMES      ;DO 12 ITERATIONS
012060  005002                  20$:    CLR     R2              ;START WITH AN ALL ZERO PATTERN
012062  012737  012102  001110          MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
012070  010204                  1$:     MOV     R2,R4           ;PUT COUNT IN R4 FOR MASK
012072  042704  000001                  BIC     #000001,R4      ;CLEAR BITS IN MASK FOR PROPER COMPARE
012076  012700  170300                  MOV     #MAPL20,R0      ;LOAD STARTING MAP REGISTER ADDRESS
012102  000240                  2$:     NOP                     ;THIS IS A SYNC POINT FOR SCOPING
012104  010210                          MOV     R2,(R0)         ;LOAD MAP REGISTER WITH COUNT PATTERN
012106  011003                          MOV     (R0),R3         ;READ MAP REGISTER INTO R3
012110  020403                          CMP     R4,R3           ;COMPARE MASK WITH DATA
012112  001402                          BEQ     3$              ;BRANCH IF DATA MATCHES
012114  004737  005352                  JSR     PC,COUNT        ;JUMP TO COUNT TO LOG ALL ERRORS
012120  005003                  3$:     CLR     R3              ;CLEAR DATA HOLDER
012122  062700  000004                  ADD     #4,R0           ;POINT TO NEXT MAP REGISTER UNDER TEST
012126  022700  170374                  CMP     #MAPL37,R0      ;SEE IF ALL REGISTERS HAVE BEEN LOADED
012132  103363                          BHIS    2$              ;BRANCH IF STILL MORE TO GO
012134  022702  177777                  CMP     #177777,R2      ;SEE IF COUNT HAS CYCLED
012140  001403                          BEQ     4$              ;BRANCH IF TEST IS DONE
012142  062702  000401                  ADD     #401,R2         ;CHANGE COUNT PATTERN
012146  000750                          BR      1$              ;CONTINUE TESTING
012150  012737  012060  001110  4$:     MOV     #20$,$LPERR     ;INITIALIZE LOOPING POINTER IN CASE
                                                                ;OF INTERMITTENT FAULTS.
012156  005737  001254                  TST     ERRCNT          ;SEE IF THERE WERE ANY ERRORS
012162  001401                          BEQ     TST15           ;BRANCH IF NO ERRORS
012164  104015                          ERROR   15              ;SUMMARY OF COUNT PATTERN FAILURES
                                                                ;IN MAPPING REGISTERS


                                ;;*****************************************************************
                                ;*TEST 15       COUNT PATTERN HIGH 20 MAP REGISTERS UPPER 6 BITS
                                ;*
                                ;*      THIS TEST WILL RUN A COUNT PATTERN THROUGH UNIBUS MAP REGISTERS
                                ;*      MAPH20 - MAPH37.  IF THE COUNT RECIEVED DOES NOT MATCH THE
                                ;*      EXPECTED PATTERN THEN THE MAP REGISTER ADDRESS, DATA RECEIVED,
                                ;*      PATTERN LOADED, AND PATTERN EXPECTED ARE REPORTED.  AT THE END
                                ;*      OF THE TEST A SUMMARY OF ALL ERRORS IS GIVEN SO THAT YOU CAN
                                ;*      DETERMINE IF THE ERROR IS BIT SENSITIVE OR REGISTER SENSITIVE.
                                ;*      THE REGISTERS ARE ON 'MAPD' AND THE OUTPUT MULTIPLEXER IS ON
                                ;*      'MAPJ'.
```

```
3002                                    ;;*****************************************************************
3003  012166                            TST15:
3004  012166  000004                           SCOPE
3005  012170  012737  012312  001324           MOV     #TST16,NXTTST   ;SAVE STARTING ADDRESS OF NEXT TEST
3006                                                                    ;FOR ESCAPE ON PARITY ERRORS
3007  012176  012737  000144  001204           MOV     #144,$TIMES     ;;DO 144 ITERATIONS
3008  012204  005002                    20$:    CLR     R2              ;START WITH AN ALL ZERO PATTERN
3009  012206  012737  012226  001110           MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
3010  012214  010204                    1$:     MOV     R2,R4           ;PUT COUNT IN R4 FOR MASK
3011  012216  042704  177700                    BIC     #177700,R4      ;CLEAR BITS IN MASK FOR PROPER COMPARE
3012  012222  012700  170302                    MOV     #MAPH20,R0      ;LOAD STARTING MAP REGISTER ADDRESS
3013  012226  000240                    2$:     NOP                     ;THIS IS A SYNC POINT FOR SCOPING
3014  012230  010210                            MOV     R2,(R0)         ;LOAD MAP REGISTER WITH COUNT PATTERN
3015  012232  011003                            MOV     (R0),R3         ;READ MAP REGISTER INTO R3
3016  012234  020403                            CMP     R4,R3           ;COMPARE MASK WITH DATA
3017  012236  001402                            BEQ     3$              ;BRANCH IF DATA MATCHES
3018  012240  004737  005352                    JSR     PC,COUNT        ;JUMP TO COUNT TO LOG ALL ERRORS
3019  012244  005003                    3$:     CLR     R3              ;CLEAR DATA HOLDER
3020  012246  062700  000004                    ADD     #4,R0           ;POINT TO NEXT MAP REGISTER UNDER TEST
3021  012252  022700  170376                    CMP     #MAPH37,R0      ;SEE IF ALL REGISTERS HAVE BEEN LOADED
3022  012256  103363                            BHIS    2$              ;BRANCH IF STILL MORE TO GO
3023  012260  022702  000177                    CMP     #177,R2 ;SEE IF COUNT HAS CYCLED
3024  012264  001403                            BEQ     4$              ;BRANCH IF TEST IS DONE
3025  012266  062702  000001                    ADD     #1,R2           ;CHANGE COUNT PATTERN
3026  012272  000750                            BR      1$              ;CONTINUE TESTING
3027  012274  012737  012204  001110    4$:     MOV     #20$,$LPERR     ;INITIALIZE LOOPING POINTER IN CASE
3028                                                                    ;OF INTERMITTENT FAULTS.
3029  012302  005737  001254                    TST     ERRCNT          ;SEE IF THERE WERE ANY ERRORS
3030  012306  001401                            BEQ     TST16           ;;BRANCH IF NO ERRORS
3031  012310  104016                            ERROR   16              ;SUMMARY OF COUNT PATTERN FAILURES
3032                                                                    ;IN MAPPING REGISTERS
3033
3034
3035                                    ;;*****************************************************************
3036                                    ;*TEST 16       CACHE REGISTER DATA PATHS
3037                                    ;*
3038                                    ;*      THIS TEST MAKES AN EFFORT TO VERIFY THE DATA PATH FROM THE
3039                                    ;*      CACHE REGISTERS THROUGH THE UNIBUS MAP TO THE CPU.  IT FIRST
3040                                    ;*      CHECKS THE CONTROL REGISTER (177746) TO SEE IF THE CACHE IS
3041                                    ;*      DISABLED, IF IT IS NOT DISABLED THEN THE CONTROL REGISTER
3042                                    ;*      IS CLEARED AND VERIFIED TO BE ZERO.  THE MAINTENENCE REGISTER
3043                                    ;*      IS ALWAYS CLEARED AND VERIFIED TO BE ZERO.  THEN THE CACHE
3044                                    ;*      ADDRESS REGISTERS (177740 & 177742) ARE READ, WITH THE ERROR
3045                                    ;*      REGISTER CLEARED,  THEY SHOULD BE 177740 AND 000003 RESPECTIVELY.
3046                                    ;*      IF ANY OF THESE CONDITIONS ARE NOT MET AN ERROR IS REPORTED
3047                                    ;*      AS BEING A POSSIBLE CACHE REGISTER DATA PATH ERROR.
3048                                    ;*      THE INPUT TO THE CACHE REGISTERS IS FROM 'MAPA' AND THE CACHE
3049                                    ;*      DATA MULTIPLEXER IS ON 'MAPH'.  THE REGISTER INPUTS ARE THE
3050                                    ;*      "X" INPPUTS, THESE GO TO 'MAPJ' TO GET BACK TO THE CPU.
3051                                    ;*
3052                                    ;;*****************************************************************
3053  012312                            TST16:
3054  012312  000004                           SCOPE
3055  012314  012737  012502  001324           MOV     #TST17,NXTTST   ;SAVE STARTING ADDRESS OF NEXT TEST
3056                                                                    ;FOR ESCAPE ON PARITY ERRORS
3057  012322  012737  012342  001110    20$:    MOV     #10$,$LPERR     ;SET LOOP ON ERROR POINTER TO 10$
```

```
3058  012330  005000                          CLR     R0              ;CLEAR R0 IN CASE CACHE IS DISABLED
3059  012332  032737  000014  177746          BIT     #14,@#CONTRL    ;SEE IF EITHER GROUP IS DISABLED
3060  012340  001007                          BNE     1$              ;BRANCH IF CACHE IS DISABLED
3061  012342  000240                  10$:    NOP                     ;THIS IS A SYNC POINT FOR SCOPING
3062  012344  005037  177746                  CLR     @#CONTRL        ;CLEAR CONTROL REGISTER
3063  012350  013700  177746                  MOV     @#CONTRL,R0     ;READ CONTROL REGISTER
3064  012354  001401                          BEQ     1$              ;BRANCH IF CONTROL REGISTER IS ZERO
3065  012356  104017                          ERROR   17              ;POSSIBLE FAULT IN CACHE REG DATA PATHS
3066  012360  012737  012366  001110  1$:     MOV     #11$,$LPERR     ;SET LOOP ON ERROR POINTER TO 11$
3067  012366  000240                  11$:    NOP                     ;THIS IS A SYNC POINT FOR SCOPING
3068  012370  005037  177750                  CLR     @#MAINT         ;CLEAR MAINTENENCE REGISTER
3069  012374  013701  177750                  MOV     @#MAINT,R1      ;READ MAINTENENCE REGISTER
3070  012400  001401                          BEQ     2$              ;BRANCH IF MAINTENENCE REGISTER IS ZERO
3071  012402  104020                          ERROR   20              ;MAINTENENCE REGISTERWILL NOT CLEAR
3072  012404  012737  012412  001110  2$:     MOV     #12$,$LPERR     ;SET LOOP ON ERROR POINTER TO 12$
3073  012412  012737  177777  177744  12$:    MOV     #-1,MEMERR      ;MAKE SURE THAT CACHE ERROR REG IS CLEAR
3074  012420  000240                          NOP                     ;THIS IS A SYNC POINT FOR SCOPING
3075  012422  013700  177740                  MOV     LOADRS,R0       ;READ CACHE LOW ADDR REG TO R0
3076  012426  022700  177740                  CMP     #177740,R0      ;SEE IF R0 = ADDR OF CACHE LOW ADDR REG
3077  012432  001401                          BEQ     3$              ;BRANCH IF DATA READ CORRECTLY
3078  012434  104021                          ERROR   21              ;CANNOT READ 177740 FROM "LOADRS"
3079  012436  012737  012444  001110  3$:     MOV     #13$,$LPERR     ;SET LOOP ON ERROR POINTER TO 13$
3080  012444  012737  177777  177744  13$:    MOV     #-1,MEMERR      ;MAKE SURE THAT CACHE ERROR REG IS CLEAR
3081  012452  000240                          NOP                     ;THIS IS A SYNC POINT FOR SCOPING
3082  012454  013700  177742                  MOV     HIADRS,R0       ;READ CACHE HIGH ADDR REG TO R0
3083  012460  042700  177700                  BIC     #177700,R0      ;MASK OFF UPPER  10 BITS
3084  012464  022700  000003                  CMP     #3,R0           ;SEE IF R0 = 3
3085  012470  001401                          BEQ     14$             ;BRANCH IF DATA MATCHES
3086  012472  104022                          ERROR   22              ;CANNOT READ 000003 FROM "HIADRS"
3087  012474  012737  012322  001110  14$:    MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
3088
3089
3090                                  ;;*****************************************************************
3091                                  ;*TEST 17       CACHE REGISTER DATA PATH COUNT PATTERN
3092                                  ;*
3093                                  ;*      THIS TEST FIRST SAVES THE CONDITION OF THE CACHE CONTROL
3094                                  ;*      REGISTER IS "$TMP5" SO THAT THE CACHE CAN BE RESTORED.
3095                                  ;*      IF EITHER BIT 3 OR 4 IS ON THE TEST BRANCHES TO "3$" TO
3096                                  ;*      TEST BITS 5 - 15 OF THE DATA PATH, IF NEITHER BIT IS ON THEN
3097                                  ;*      THE TEST RUNS A COUNT PATTERN THRU THE CONTROL REGISTER FROM
3098                                  ;*      00 - 77.   IF ANY BITS ARE DETECTED AS BEING STUCK THE ERROR
3099                                  ;*      IS REPORTED AND A SUMMARY OF THOSE ERRORS IS GIVEN.  THEN THE
3100                                  ;*      TEST FORCES MISSES IN BOTH GROUPS OF THE CACHE AND PROCEEDS
3101                                  ;*      TO RUN A COUNT PATTERN THROUGH THE CACHE MAINTENENCE REGISTER
3102                                  ;*      (177746) FROM 000000 - 177760 IN INCREMENTS OF 20.
3103                                  ;*      THE CODE IS SET UP IN SUCH A WAY THAT NO PARITY ERRORS
3104                                  ;*      SHOULD BE GENERATED.  AGAIN ALL BAD MATCHES OF DATA WILL BE
3105                                  ;*      REPORTED AND A SUMMARY OF ALL ERRORS IS GIVEN AT THE END OF
3106                                  ;*      THE TEST AND THE CACHE IS RESTORED TO ITS PREVIOUS CONDITION.
3107                                  ;*
3108                                  ;;*****************************************************************
3109  012502                          TST17:
3110  012502  000004                          SCOPE
3111  012504  012737  013124  001324          MOV     #TST20,NXTTST   ;SAVE STARTING ADDRESS OF NEXT TEST
3112                                                                  ;FOR ESCAPE ON PARITY ERRORS
3113  012512  012737  000012  001204          MOV     #12,$TIMES      ;;DO 12 ITERATIONS
```

```
                                                    TBITO                   ;TURN OFF T BIT TRAPPING IF ON
012520  104416
012522  013737  177746  001202  20$:      MOV     @#CONTRL,$TMP5          ;SAVE CONTROL REG TO RESTORE CACHE
012530  032737  000014  177746             BIT     #14,@#CONTRL            ;SEE IF EITHER GROUP IS DISABLED
012536  001047                             BNE     3$                      ;SKIP FIRST PART OF TEST IF DISABLED
012540  012700  177746                     MOV     #CONTRL,R0              ;LOAD ADDR OF CONTROL REG INTO R0
012544  005002                             CLR     R2                      ;START COUNT PATTERN AT ZERO
012546  012737  012554  001110             MOV     #1$,$LPERR              ;SET LOOP ON ERROR POINTER TO 1$
012554  000240                     1$:     NOP                             ;THIS IS A SYNC POINT FOR SCOPING
012556  010210                             MOV     R2,(R0)                 ;LOAD COUNT INTO CONTROL REG
012560  011003                             MOV     (R0),R3                 ;READ COUNT FROM CONTROL REG
012562  020203                             CMP     R2,R3                   ;SEE IF COUNT = DATA READ
012564  001402                             BEQ     2$                      ;BRANCH IF DATA MATCHES
012566  004737  005516                     JSR     PC,CACOUNT              ;LOG AND REPORT COUNT PATTERN ERROR
012572  005202                     2$:     INC     R2                      ;CHANGE COUNT PATTERN
012574  022702  000077                     CMP     #77,R2                  ;SEE IF COUNT HAS PASSED MAXIMUM
012600  103365                             BHIS    1$                      ;BRANCH IF COUNT HASN'T CYCLED
012602  005737  001254                     TST     ERRCNT                  ;SEE IF ANY ERRORS ON THIS CYCLE
012606  001423                             BEQ     3$                      ;BRANCH IF NO ERRORS
012610  104023                             ERROR   23                      ;COUNT THRU CONTROL REG FAILED
012612  005037  001322                     CLR     RETRY                   ;CLEAR RETRY FLAG AN THE START OF
                                                                           ;EACH TEST
012616  005037  001254                     CLR     ERRCNT                  ;CLEAR THE MULTIPLE ERROR COUNTER
012622  005037  001232                     CLR     DATAOR                  ;LOCATION FOR LOGICAL OR OF BAD DATA
012626  005037  001226                     CLR     ADDROR                  ;LOCATION FOR LOGICAL OR OF ADDRESS
012632  005037  001236                     CLR     PATTOR                  ;LOCATION FOR LOGICAL OR OF PATTERN LOADED
012636  012700  177777                     MOV     #-1,R0                  ;LOAD -1 INTO R0 TO INITIALIZE LOGICAL AND LOCS
012642  010037  001230                     MOV     R0,DATAND               ;LOCATION FOR LOGICAL AND OF BAD DATA
012646  010037  001224                     MOV     R0,ADRAND               ;LOCATION FOR LOGICAL AND OF ADDRESS
012652  010037  001234                     MOV     R0,PATAND               ;LOCATION FOR LOGICAL AND OF PATTERN LOADED
012656  012737  000014  177746     3$:     MOV     #14,@#CONTRL            ;DISABLE BOTH GROUPS OF CACHE
012664  012700  177750                     MOV     #MAINT,R0               ;LOAD ADDR OF MAINTENENCE REG INTO R0
012670  010001                             MOV     R0,R1                   ;R1 HAS #MAINT SO CLR (R1) HAS P.B.'S ON
012672  005002                             CLR     R2                      ;START COUNT PATTERN AT ZERO
012674  012737  012702  001110             MOV     #4$,$LPERR              ;SET LOOP ON ERROR POINTER TO 4$
012702  010205                     4$:     MOV     R2,R5                   ;PUT COUNT IN R5 FOR CORRECT PARITY
012704  000240                             NOP                             ;THIS IS A SYNC POINT FOR SCOPING
012706  010510                             MOV     R5,(R0)                 ;LOAD MAINT REGISTER INST HAS P.B.'S ON
012710  011003                             MOV     (R0),R3                 ;READ MAINT REG (P.B.'S ON)
012712  005011                             CLR     (R1)                    ;TURN OFF MAINT REG (P.B.'S ON)
012714  050000                             BIS     R0,R0                   ;DUMMY INST WITH P.B.'S ON
012716  020203                             CMP     R2,R3                   ;SEE IF COUNT = DATA READ
012720  001402                             BEQ     5$                      ;BRANCH IF DATA MATCHES
012722  004737  005516                     JSR     PC,CACOUNT              ;LOG AND REPORT COUNT PATTERN ERROR
012726  062702  000020             5$:     ADD     #20,R2                  ;CHANGE COUNT PATTERN
012732  001363                             BNE     4$                      ;BRANCH IF COUNT HAS NOT PASSED 177760
012734  005737  001254                     TST     ERRCNT                  ;SEE IF ANY ERRORS ON THIS COUNT CYCLE
012740  001401                             BEQ     19$                     ;BRANCH IF NO ERRORS ON THIS PART
012742  104024                             ERROR   24                      ;SUMMARY OF COUNT ERRORS IN MAINT REG
012744  012737  012522  001110     19$:    MOV     #20$,$LPERR             ;SET LOOP POINTER TO START OF TEST
012752  013737  001202  177746             MOV     $TMP5,@#CONTRL          ;RETURN CACHE TO PREVIOUS STATE
012760  000004                             SCOPE                           ;

                                    ;*      AT THIS POINT 22-BIT RELOCATION FROM MEMORY MANAGEMENT
                                    ;*      IS ENABLED, WITH THE KIPAR'S MAPPED TO PHYSICAL 0-24K.
                                    ;*      KIPAR6 IS MAPPED TO THE UNIBUS (170000) AND
                                    ;*      KIPAR7 IS MAPPED TO THE I/O PAGE (177600).
```

```
3170
3171  012762  104420                          TBITR                   ;RESTORE THE T BIT TO ITS CONDITION
3172                                                                  ;BEFORE THE LAST TEST
3173  012764  012700  077406                  MOV     #77406,R0       ;MAKE THE KERNEL I-SPACE PAGES ALL
3174                                                                  ;4K, UPWARD EXPANDABLE, READ/WRITE
3175  012770  010037  172300                  MOV     R0,KIPDR0       ;KERNEL I-SPACE PAGE 0
3176  012774  010037  172302                  MOV     R0,KIPDR1       ;KERNEL I-SPACE PAGE 1
3177  013000  010037  172304                  MOV     R0,KIPDR2       ;KERNEL I-SPACE PAGE 2
3178  013004  010037  172306                  MOV     R0,KIPDR3       ;KERNEL I-SPACE PAGE 3
3179  013010  010037  172310                  MOV     R0,KIPDR4       ;KERNEL I-SPACE PAGE 4
3180  013014  010037  172312                  MOV     R0,KIPDR5       ;KERNEL I-SPACE PAGE 5
3181  013020  010037  172314                  MOV     R0,KIPDR6       ;KERNEL I-SPACE PAGE 6
3182  013024  010037  172316                  MOV     R0,KIPDR7       ;KERNEL I-SPACE PAGE 7
3183  013030  012737  000000  172340          MOV     #000,KIPAR0     ;MAP TO PHYSICAL 0
3184  013036  012737  000200  172342          MOV     #200,KIPAR1     ;MAP TO PHYSICAL 4K - 8K
3185  013044  012737  000400  172344          MOV     #400,KIPAR2     ;MAP TO PHYSICAL 8K - 12K
3186  013052  012737  000600  172346          MOV     #600,KIPAR3     ;MAP TO PHYSICAL 12K - 16K
3187  013060  012737  001000  172350          MOV     #1000,KIPAR4    ;MAP TO PHYSICAL 16K - 20K
3188  013066  012737  001200  172352          MOV     #1200,KIPAR5    ;MAP TO PHYSICAL 20K - 24K
3189  013074  012737  170000  172354          MOV     #170000,KIPAR6  ;MAP TO UNIBUS
3190  013102  012737  177600  172356          MOV     #177600,KIPAR7  ;MAP TO I/O PAGE
3191  013110  012737  000001  177572          MOV     #BIT0,MMR0      ;ENABLE FULL 18-BIT MAPPING
3192  013116  012737  000020  172516          MOV     #BIT4,MMR3      ;ENABLE 22-BIT MAPPING
3193
3194                                  ;;**************************************************************
3195                                  ;*TEST 20       MAP REGISTER ADDRESS DECODE TEST
3196                                  ;*
3197                                  ;*      THIS TEST TRIES TO VERIFY THAT NONE OF THE INPUTS TO THE ADDRESS
3198                                  ;*      DECODER FOR THE UNIBUS MAP REGISTERS ON 'MAPB' IS STUCK TRUE.
3199                                  ;*      KIPAR6 IS SETUP TO HOLD 177702 AND R4 HAS THE VIRTUAL ADDRESS
3200                                  ;*      TO SELECT MAPLOO, THRU KIPAR6.  THE TEST THEN CHANGES ONE BIT
3201                                  ;*      AT A TIME IN PAR6 SO THAT IT SHOULD NEVER REFERENCE MAPLOO.  IF
3202                                  ;*      IT DOES AN ERROR IS REPORTED.
3203                                  ;*
3204                                  ;;**************************************************************
3205  013124                          TST20:
3206  013124  012737  013154  001106          MOV     #20$,$LPADR     ;SET LOOP ON TEST POINTER TO 20$
3207  013132  012737  013154  001110          MOV     #20$,$LPERR     ;SET LOOP ON ERROR POINTER TO 20$
3208  013140  012737  000020  001102          MOV     #20,$TSTNM      ;SETUP TEST NUMBER AND CLR ERROR FLAG
3209  013146  013737  001102  177570          MOV     $TSTNM,DISPLAY  ;DISPLAY TEST NUMBER FOR ALL TO SEE
3210                                          .EQUIV  BIT4,TIMOUT     ;BIT4 IS TIMEOUT BIT IN CPU ERROR REG
3211  013154  012737  000020  001264  20$:   MOV     #TIMOUT,CPUEXP  ;EXPECTING CPU TIME OUT ON UNIBUS
3212  013162  012737  013222  001110          MOV     #10$,$LPERR     ;SET LOOP ON ERROR POINTER TO 10$
3213  013170  012737  177702  172354          MOV     #177702,KIPAR6  ;PUT MAP REG 0 ADDR IN PAR6
3214  013176  012702  175254                  MOV     #175254,R2      ;PATTERN FOR TESTING.
3215  013202  010237  170200                  MOV     R2,@#MAPLO      ;LOAD MAP REGISTER 0
3216  013206  012700  004000                  MOV     #BIT11,R0       ;SET BIT 11 TO FLOAT THRU PAR6
3217  013212  012704  140000                  MOV     #140000,R4      ;VIRT.ADDR. TO SELECT PAR6
3218  013216  074037  172354          1$:     XOR     R0,KIPAR6       ;CHANGE A BIT OF MAP REG 0'S ADDR
3219  013222  000240                  10$:    NOP                     ;THIS IS A SYNC POINT FOR SCOPING
3220  013224  011401                          MOV     (R4),R1         ;READ LOCATION POINTED TO BY PAR6
3221  013226  020201                          CMP     R2,R1           ;SEE IF DATA FETCHED MATCHES PATTERN
3222  013230  001007                          BNE     3$              ;BRANCH IF NOT SAME
3223  013232  012714  000000                  MOV     #0,(R4)         ;TRY TO CLEAR THIS LOCATION
3224  013236  005737  170200                  TST     @#MAPLO         ;SEE IF MAP REG 0 GOT CLEARED
3225  013242  001001                          BNE     2$              ;BRANCH IF MAP REG NOT ZERO
```

```
                                        ERROR   25              ;GOT TO MAPLO WITH ONE BIT
                                                                ;DIFFERENT IN ADDRESS FROM 770200
                                2$:     MOV     R1,(R4)         ;RELOAD THE LOCATION
                                3$:     XOR     R0,KIPAR6       ;RESTORE BIT TO ORIGINAL STATUS
                                        ASR     R0              ;RIGHT SHIFT ONE PLACE
                                        CMP     R0,#1           ;SEE IF TEST IS OVER
                                        BNE     1$              ;BRANCH TO CONTINUE TEST
                                        MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
                                        CLR     CPUEXP          ;ZERO EXPECTED CPU TRAP FLAG

                        ;;**************************************************************
                        ;*TEST 21       CACHE REGISTER ADDRESS DECODE TEST
                        ;*
                        ;*      THIS TEST TRIES TO VERIFY THAT NONE OF THE INPUTS TO THE ADDRESS
                        ;*      DECODER FOR THE CACHE REGISTERS ON 'MAPB' IS STUCK TRUE.
                        ;*      KIPAR6 IS SETUP TO HOLD 177740 AND R4 HAS THE VIRTUAL ADDRESS
                        ;*      TO SELECT 'LOADRS' THRU KIPAR6.  THE TEST THEN CHANGES ONE BIT
                        ;*      AT A TIME IN PAR6 SO THAT IT SHOULD NEVER REFERENCE 'LOADRS'.
                        ;*      IF IT DOES AN ERROR IS REPORTED.
                        ;*
                        ;;**************************************************************
                        TST21:
                                SCOPE
                                MOV     #TST22,NXTTST   ;SAVE STARTING ADDRESS OF NEXT TEST
                                                        ;FOR ESCAPE ON PARITY ERRORS
                        20$:    MOV     #TIMOUT,CPUEXP  ;EXPECTING UNIBUS TIMEOUT IN THIS TEST
                                MOV     #10$,$LPERR     ;SET LOOP ON ERROR POINTER TO 10$
                                MOV     #177777,KIPAR6  ;PUT CACHE REG BASE IN PAR6
                                MOV     #177740,R2      ;PUT DATA PATTERN IN R2
                                MOV     #BIT5,R0        ;SET BIT 5 TO FLOAT THRU PAR6
                                MOV     #140040,R4      ;VIRT. ADDR. TO SELECT PAR6
                        1$:     XOR     R0,KIPAR6       ;CHANGE A BIT OF CACHE REG'S ADDR
                        10$:    NOP                     ;THIS IS A SYNC POINT FOR SCOPING
                                MOV     (R4),R1         ;TRY TO READ CACHE 'LOADRS' WITH
                                                        ;ONE BIT DIFFERENT THAN ADDR. 777740
                                CMP     R2,R1           ;SEE IF YOU GOT CACHE 'LOADRS'
                                BNE     2$              ;BRANCH TO CONTINUE TEST.
                                ERROR   26              ;READ CACHE 'LOADRS' WITHOUT ADRS 777740
                        2$:     XOR     R0,KIPAR6       ;RESTORE BIT TO ORIGINAL STATUS.
                                ASR     R0              ;RIGHT SHIFT R0 1 PLACE
                                BNE     1$              ;BRANCH IF NOT 0
                                MOV     #11$,$LPERR     ;SET LOOP ON ERROR POINTER TO 11$
                                MOV     #177700,R1      ;PUT ADDRESS IN R1
                        11$:    NOP                     ;THIS IS A SYNC POINT FOR SCOPING
                                MOV     (R1),R3         ;TRY TO READ 177700
                                CMP     R2,R3           ;SEE IF YOU GET 177740
                                BNE     3$              ;BRANCH IF YOU DON'T
                                ERROR   27              ;READ CACHE 'LOADRS' WITHOUT ADRS 777740
                        3$:     MOV     #12$,$LPERR     ;SET LOOP ON ERROR POINTER TO 12$
                                MOV     #177760,R1      ;PUT ADDRESS IN R1
                        12$:    NOP                     ;THIS IS A SYNC POINT FOR SCOPING
                                MOV     (R1),R3         ;TRY TO READ 177760
                                CMP     R2,R3           ;SEE IF YOU GET 177740
                                BNE     4$              ;BRANCH IF YOU DON'T
                                ERROR   27              ;READ CACHE 'LOADRS' WITHOUT ADRS 777740
                        4$:     MOV     #13$,$LPERR     ;SET LOOP ON ERROR POINTER TO 13$
```

```
3282  013450  012701  177754            MOV     #177754,R1      ;PUT ADDRESS IN R1
3283  013454  000240            13$:    NOP                     ;THIS IS A SYNC POINT FOR SCOPING
3284  013456  011103                    MOV     (R1),R3         ;TRY TO READ 177754
3285  013460  020203                    CMP     R2,R3           ;SEE IF YOU GET 177740
3286  013462  001001                    BNE     19$             ;BRANCH IF YOU DIDN'T READ ADDRS 177740
3287  013464  104027                    ERROR   27              ;READ CACHE 'LOADRS' WITHOUT ADRS 777740
3288  013466  012737  013306  001110  19$:    MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
3289  013474  005037  001264            CLR     CPUEXP          ;ZERO EXPECTED CPU TRAP CONDITION
3290
3291
3292                                    ;;*****************************************************************
3293                                    ;*TEST 22       DATA PATH, UNIBUS TO MAIN MEMORY
3294                                    ;*
3295                                    ;*      THIS TEST RUNS A COUNT PATTERN THROUGH A MEMORY LOCATION VIA
3296                                    ;*      THE UNIBUS.  THE UNIBUS MAP IS LEFT OFF DURING THIS TEST SO
3297                                    ;*      THAT THE ADDRESS IS NOT RELOCATED.  THE TEST TRIES TO LOAD THE
3298                                    ;*      PATTERN INTO ADDRESS 040000 (8K) BUT IF THE MAP JUMPERS ARE
3299                                    ;*      SET NOT TO RESPOND TO THAT ADDRESS THE NEXT 4K IS TRIED UNTIL
3300                                    ;*      THE TEST GETS TO MAIN MEMORY FROM THE UNIBUS.  IF THIS TEST
3301                                    ;*      DETERMINES THAT IT CANNOT GET TO MAIN MEMORY FROM THE UNIBUS
3302                                    ;*      IT REPORTS THE FACT AND SKIPS THE NEXT TEST FOR VERIFICATION.
3303                                    ;*      THE DATA PATH TO MAIN MEMORY FROM THE UNIBUS IS ON 'MAPA' AND
3304                                    ;*      THE PATH FROM MAIN MEMORY TO THE UNIBUS IS FROM THE CACHE
3305                                    ;*      "DTML CDMX D00" - "DTML CDMX D15" INTO 'MAPH' THE 'Z' INPUTS
3306                                    ;*      TO THE MULTIPLEXER AND THEN TO 'MAPJ' AND OUT TO THE UNIBUS.
3307                                    ;*
3308                                    ;;*****************************************************************
3309  013500                    TST22:
3310  013500  000004                    SCOPE
3311  013502  012737  013732  001324    MOV     #TST23,NXTTST   ;SAVE STARTING ADDRESS OF NEXT TEST
3312                                                            ;FOR ESCAPE ON PARITY ERRORS
3313  013510  012737  000012  001204    MOV     #12.,$TIMES     ;;DO 12 ITERATIONS
3314  013516  004737  005054    20$:    JSR     PC,CLRMAP       ;CLEAR ALL MAP REGISTERS
3315  013522  012737  000020  001264    MOV     #TIMOUT,CPUEXP  ;TIMEOUTS MIGHT OCCUR IN THIS TEST.
3316  013530  012737  013536  001110    MOV     #10$,$LPERR     ;SET LOOP ON ERROR POINTER TO 10$
3317  013536  012737  170400  172354  10$:    MOV     #170400,KIPAR6  ;START WITH ADDRESS 8K FROM UNIBUS
3318  013544  005037  001266    1$:     CLR     PCPUER          ;CLEAR ERROR CONDITION LOCATION
3319  013550  000240                    NOP                     ;THIS IS A SYNC POINT FOR SCOPING
3320  013552  013700  140000            MOV     @#140000,R0     ;TRY TO READ ADDRESS POINTED TO BY PAR6
3321  013556  005737  001266            TST     PCPUER          ;SEE IF READ OF ADDRESS TIMED OUT
3322  013562  001412                    BEQ     2$              ;BRANCH IF REFERENCE WAS GOOD
3323  013564  062737  000200  172354  4$:     ADD     #200,KIPAR6     ;TRY NEXT 4K BLOCK OF MEMORY
3324  013572  022737  177600  172354    CMP     #177600,KIPAR6 ;SEE IF YOU'VE POINTED TO I/O PAGE
3325  013600  001361                    BNE     1$              ;BRANCH IF NOT
3326  013602  104030                    ERROR   30              ;NO UNIBUS ADDRESSES RESPOND
3327  013604  000137  014062            JMP     SIZEJ           ;JUMP TO SIZE JUMPER TEST
3328  013610  013737  172354  172352  2$:     MOV     KIPAR6,KIPAR5   ;PUT PAR6 INTO PAR5
3329  013616  042737  170000  172352    BIC     #170000,KIPAR5  ;MAKE PAR5 A NON UNIBUS ADDRESS
3330  013624  012737  173214  120000    MOV     #173214,@#120000 ;PUT RANDOM NUMBER INTO TEST LOCATION BY FAST BUS
3331  013632  013701  140000            MOV     @#140000,R1     ;READ TEST LOCATION VIA UNIBUS
3332  013636  022701  173214            CMP     #173214,R1      ;SEE IF DATA WAS READ PROPERLY
3333  013642  001403                    BEQ     3$              ;DATA OKAY NOW VERIFY DATA PATH
3334  013644  020001                    CMP     R0,R1           ;SEE IF DATA CHANGED FROM FIRST READ
3335  013646  001001                    BNE     3$              ;BRANCH IF DATA CHANGED
3336  013650  000745                    BR      4$              ;TRY NEXT 4K BLOCK
3337  013652  005001            3$:     CLR     R1              ;CLEAR REGISTER TO HOLD COUNT
```

```
3338  013654  012737  013666  001110          MOV     #11$,$LPERR     ;SET LOOP ON ERROR POINTER TO 11$
3339  013662  012702  140000                  MOV     #140000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3340  013666  000240                  11$:    NOP                     ;THIS IS A SYNC POINT FOR SCOPING
3341  013670  010112                  5$:     MOV     R1,(R2)         ;LOAD COUNT INTO TEST LOCATION VIA U.B.
3342  013672  011200                          MOV     (R2),R0         ;READ TEST LOCATION BACK VIA UNIBUS
3343  013674  020100                          CMP     R1,R0           ;COMPARE COUNT WITH DATA READ
3344  013676  001402                          BEQ     6$              ;BRANCH IF DATA MATCHES
3345  013700  004737  005442                  JSR     PC,UBCOUNT      ;COUNT FAILED REPORT ERROR
3346  013704  005201                  6$:     INC     R1              ;INCREASE COUNT
3347  013706  001370                          BNE     5$              ;BRANCH IF COUNT HASN'T CYCLED
3348  013710  005737  001254                  TST     ERRCNT          ;WERE THERE ANY ERRORS ON THIS TEST
3349  013714  001401                          BEQ     19$             ;BRANCH IF NO ERRORS ON THIS TEST
3350  013716  104031                          ERROR   31              ;SUMMARY OF ERRORS ON THE UNIBUS
3351                                                                  ;DATA PATH
3352  013720  012737  013516  001110  19$:    MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
3353  013726  005037  001264                  CLR     CPUEXP          ;ZERO EXPECTED CPU TRAP CONDITION
3354
3355
3356                                  ;;*****************************************************************
3357                                  ;*TEST 23       MAP DOESN'T RELOCATE IF NOT ENABLED
3358                                  ;*
3359                                  ;*      THIS TEST VERIFIES THAT THE UNIBUS MAP DOES NOT RELOCATE IF BIT5
3360                                  ;*      OF MMR3 IS NOT SET.  THE TEST ASSUMES THAT THE PREVIOUS TEST HAS
3361                                  ;*      RUN SUCCESSFULLY AND LEFT KIPAR6 POINTING TO THE FIRST UNIBUS
3362                                  ;*      MAPPING REGISTER THAT THE UNIBUS MAP WILL RESPOND TO GREATER
3363                                  ;*      THAN OR EQUAL TO MAPREG #2.  KIPAR5 IS ALSO POINTING TO THE
3364                                  ;*      SAME MEMORY BASE ADDRESS EXCEPT IT POINTS OVER THE FASTBUS.
3365                                  ;*      THE TEST THEN SETS ONE BIT IN EACH A.L.U. OF THE UNIBUS MAP
3366                                  ;*      AND TRIES TO REFERENCE MAIN MEMORY OVER THE UNIBUS.  SINCE THE
3367                                  ;*      MAP IS NOT ENABLED THE LOAD WILL GO TO MAIN MEMORY UNRELOCATED.
3368                                  ;*
3369                                  ;*****************************************************************
3370  013732                          TST23:
3371  013732  000004                          SCOPE
3372  013734  012737  014044  001324          MOV     #TST24,NXTTST   ;SAVE STARTING ADDRESS OF NEXT TEST
3373                                                                  ;FOR ESCAPE ON PARITY ERRORS
3374  013742  012737  000020  001264  20$:    MOV     #TIMOUT,CPUEXP  ;MIGHT TIME OUT OVER UNIBUS
3375  013750  012737  014006  001110          MOV     #10$,$LPERR     ;SET LOOP ON ERROR POINTER TO 10$
3376  013756  013700  172354                  MOV     KIPAR6,R0       ;PUT UNIBUS ADDRESS OF MAP REG IN R0
3377  013762  072027  177773                  ASH     #-5,R0          ;RIGHT SHIFT R0 5 PLACES
3378  013766  042700  007400                  BIC     #007400,R0      ;STRIP OFF EXTRANEOUS BITS.
3379  013772  012720  021042                  MOV     #021042,(R0)+   ;SET BOTTOM BIT IN EACH ALU
3380  013776  012710  000042                  MOV     #42,(R0)        ;SET BOTTOM BIT IN EACH ALU
3381  014002  005037  120000                  CLR     @#120000        ;CLEAR TEST LOCATION VIA FAST BUS
3382  014006  000240                  10$:    NOP                     ;THIS IS A SYNC POINT FOR SCOPING
3383  014010  012737  043207  140000          MOV     #43207,@#140000 ;LOAD TEST LOCATION VIA UNIBUS
3384                                                                  ;THIS LOAD SHOULD NOT BE RELOCATED
3385                                                                  ;BY THE UNIBUS MAP, SINCE BIT05 OF
3386                                                                  ;MMR3 IS CLEAR.
3387  014016  013703  120000                  MOV     @#120000,R3     ;READ TEST LOCATION VIA FAST BUS
3388  014022  022703  043207                  CMP     #43207,R3       ;SEE IF DATA MATCHES
3389  014026  001401                          BEQ     1$              ;BRANCH IF DATA GOOD
3390  014030  104032                          ERROR   32              ;MAP RELOCATED WHEN NOT ENABLED
3391  014032  012737  013742  001110  1$:     MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
3392  014040  005037  001264                  CLR     CPUEXP          ;ZERO EXPECTED CPU TRAP CONDITION
3393
```

```
3394
3395                                    ;;*****************************************************************
3396                                    ;*TEST 24       SIZE JUMPER LOCATION
3397                                    ;*
3398                                    ;*      THIS TEST DETERMINES THE SETTING OF THE JUMPERS ON THE UNIBUS
3399                                    ;*      MAP WHICH ALLOW THE MAP TO RESPOND TO THOSE ADDRESSES BETWEEN
3400                                    ;*      THE JUMPER RANGE.  THE DEFAULT SETTING ALLOWS THE MAP TO RESPOND
3401                                    ;*      TO ADDRESSES 000000 - 757776 ON THE UNIBUS.  IF THE JUMPERS ARE
3402                                    ;*      NOT SET IN THEIR DEFAULT POSITION AN ERROR MESSAGE IS GIVEN, AND
3403                                    ;*      THE NUMBER THAT IS REMOVED BY THE JUMPER SETTING IS COMPARED WITH
3404                                    ;*      THE NUMBER THAT SHOULD NOT RESPOND.  IF THESE NUMBERS DON'T
3405                                    ;*      CORRESPOND THE ERROR COULD BE IN THE COMPARE CIRCUIT ON 'MAPX'.
3406                                    ;*
3407                                    ;*****************************************************************
3408  014044                            TST24:
3409  014044  000004                           SCOPE
3410  014046  012737  014500  001324           MOV     #TST25,NXTTST   ;SAVE STARTING ADDRESS OF NEXT TEST
3411                                                                    ;FOR ESCAPE ON PARITY ERRORS
3412  014054  012737  000001  001204           MOV     #1,$TIMES       ;;DO 1 ITERATION
3413  014062                            SIZEJ:
3414  014062  012737  014112  001106           MOV     #20$,$LPADR     ;SET LOOP ON TEST POINTER TO 20$
3415  014070  012737  014112  001110           MOV     #20$,$LPERR     ;SET LOOP ON ERROR POINTER TO 20$
3416  014076  012737  000024  001102           MOV     #24,$TSTNM      ;SETUP TEST NUMBER AND CLR ERROR FLAG
3417  014104  013737  001102  177570           MOV     $TSTNM,DISPLAY  ;DISPLAY TEST NUMBER FOR ALL TO SEE
3418  014112  012737  000020  001264  20$:      MOV     #TIMOUT,CPUEXP  ;EXPECTING CPU TIME OUT ON UNIBUS
3419  014120  012700  170200                   MOV     #MAPLO,R0       ;LOAD ADDRESS OF FIRST MAP REG
3420  014124  012720  020000           1$:     MOV     #20000,(R0)+    ;LOAD 4K INTO ALL MAP REGISTERS
3421  014130  005020                           CLR     (R0)+           ;INSURE THAT ALL REGS HAVE UPPER BITS CLR
3422  014132  022700  170374                   CMP     #MAPL37,R0      ;SEE IF LAST REG IS LOADED
3423  014136  101372                           BHI     1$              ;BRANCH IF THERE ARE MORE TO LOAD
3424  014140  052737  000040  172516           BIS     #BIT5,@#MMR3    ;TURN ON MAP RELOCATION
3425  014146  012700  117776                   MOV     #117776,R0      ;THIS WILL BE USED TO SELECT PAR 4
3426  014152  012737  170000  172350           MOV     #170000,KIPAR4  ;WE START TESTING WITH MAP 0
3427  014160  012701  000200                   MOV     #200,R1         ;CONSTANT USED TO ADD TO PAR 4
3428  014164  012702  125252                   MOV     #125252,R2      ;CONSTANT TO LOAD INTO LOCATION 37776
3429  014170  005037  037776           2$:     CLR     @#37776         ;CLEAR TEST LOCATION
3430  014174  010210                           MOV     R2,(R0)         ;TRY TO LOAD TEST CELL THROUGH MAP
3431  014176  023702  037776                   CMP     @#37776,R2      ;SEE IF TEST LOCATION WAS LOADED
3432  014202  001411                           BEQ     3$              ;BRANCH IF IT WAS LOADED
3433  014204  060137  172350                   ADD     R1,KIPAR4       ;CELL NOT LOADED, TEST NEXT MAP REG
3434  014210  022737  177600  172350           CMP     #177600,KIPAR4  ;SEE IF YOU'RE POINTING TO I/O PAGE
3435  014216  001364                           BNE     2$              ;GO TYPE NEXT MAP REGISTER
3436  014220  104033                           ERROR   33              ;FATAL ERROR RESTARTING PROGRAM
3437  014222  000137  010000                   JMP     START           ;RESTART PROGRAM
3438  014226  013737  172350  001240  3$:      MOV     KIPAR4,LOWEST   ;FOUND THE LOWEST USABLE MAP REG
3439  014234  013737  172350  001242  4$:      MOV     KIPAR4,HIGEST   ;THIS WILL END UP BEING LAST USABLE REG
3440  014242  060137  172350                   ADD     R1,KIPAR4       ;TRY NEXT MAP REG TO SEE IF IT RESPONDS
3441  014246  022737  177600  172350           CMP     #177600,KIPAR4  ;SEE IF ALL MAP REGS HAVE BEEN TRIED
3442  014254  001433                           BEQ     7$              ;BRANCH IF ALL ARE DONE
3443  014256  005037  037776                   CLR     @#37776         ;CLEAR TEST LOCATION
3444  014262  010210                           MOV     R2,(R0)         ;TRY TO LOAD TEST CELL THROUGH THE MAP
3445  014264  023702  037776                   CMP     @#37776,R2      ;SEE IF TEST LOCATION WAS LOADED
3446  014270  001761                           BEQ     4$              ;BRANCH IF IT WAS LOADED
3447  014272  005037  037776           5$:     CLR     @#37776         ;CLEAR TEST LOCATION
3448  014276  005237  001254                   INC     ERRCNT          ;COUNT OF REGS AFTER LAST ONE USABLE
3449  014302  010210                           MOV     R2,(R0)         ;TRY TO LOAD TEST CELL THRU MAP
```

```
3450  014304  023702  037776                  CMP     @#37776,R2      ;SEE IF TEST LOCATION WAS LOADED
3451  014310  001402                          BEQ     6$              ;BRANCH IF LOCATION WAS LOADED
3452  014312  005237  001256                  INC     CNTR            ;COUNT OF REGS THAT FAILED TO RESPOND
3453  014316  060137  172350          6$:     ADD     R1,KIPAR4       ;POINT TO NEXT MAP REG UNDER TEST
3454  014322  022737  177600  172350          CMP     #177600,KIPAR4  ;SEE IF TEST IS OVER
3455  014330  001360                          BNE     5$              ;BRANCH IF TEST NOT DONE
3456  014332  023737  001254  001256          CMP     ERRCNT,CNTR     ;SEE IF TRIES = FAILURES
3457  014340  001401                          BEQ     7$              ;BRANCH IF EQUAL
3458  014342  104034                          ERROR   34              ;MAP JUMPER COMPARE CIRCUIT BAD
3459  014344  005037  001254          7$:     CLR     ERRCNT          ;CLEAR TRIES COUNTER
3460  014350  005037  001256                  CLR     CNTR            ;CLEAR FAILURE COUNTER
3461  014354  005037  001264                  CLR     CPUEXP          ;NO CPU TRAPS EXPECTED IN NEXT TEST
3462  014360  005737  001100                  TST     $PASS           ;SEE IF THIS IS FIRST PASS
3463  014364  001045                          BNE     TST25           ;;GO TO NEXT TEST IF NOT THE FIRST PASS
3464  014366  023727  001240  170000          CMP     LOWEST,#170000  ;SEE IF LOWER JUMPER IS DEFAULT
3465  014374  001004                          BNE     8$              ;BRANCH IF NOT.
3466  014376  023727  001242  177400          CMP     HIGEST,#177400  ;SEE IF UPPER JUMPER IS DEFAULT.
3467  014404  001401                          BEQ     9$              ;BRANCH IF JUMPERS DEFAULT.
3468  014406  104035                  8$:     ERROR   35              ;MAP SIZE JUMPERS NOT DEFAULT
3469                                  ;;
3470                                  ;*      SETUP POINTERS TO THE LOWEST AND THE HIGHEST USEABLE
3471                                  ;*      MAPPING REGISTERS TO CONTINUE WITH TEST.
3472                                  ;;
3473  014410  013700  001242          9$:     MOV     HIGEST,R0
3474  014414  013701  001240                  MOV     LOWEST,R1
3475  014420  042700  170000                  BIC     #170000,R0
3476  014424  042701  170000                  BIC     #170000,R1
3477  014430  072027  177773                  ASH     #-5,R0          ;RIGHT SHIFT R0 5 PLACES
3478  014434  072127  177773                  ASH     #-5,R1          ;RIGHT SHIFT R1 5 PLACES
3479  014440  062701  170200                  ADD     #170200,R1
3480  014444  062700  170200                  ADD     #170200,R0
3481  014450  010137  001244                  MOV     R1,LREGL
3482  014454  062701  000002                  ADD     #2,R1           ;POINT TO UPPER BITS OF MAP REG
3483  014460  010137  001246                  MOV     R1,LREGU
3484  014464  010037  001250                  MOV     R0,HREGL
3485  014470  062700  000002                  ADD     #2,R0           ;POINT TO UPPER BITS OF MAP REG
3486  014474  010037  001252                  MOV     R0,HREGU
3487
3488
3489
3490                                  ;;**************************************************************
3491                                  ;*TEST 25       ENSURE THAT THERE IS NO DUAL MAPPING
3492                                  ;*
3493                                  ;*      THIS TEST VERIFIES THAT THE OTHER INPUT (X) TO THE ADDRESS
3494                                  ;*      MULTIPLEXER ON 'MAPC' IS FUNCTIONING PROPERLY.  IT CLEARS
3495                                  ;*      ALL THE MAP REGISTERS EXCEPT THE ONE UNDER TEST, AND LOADS
3496                                  ;*      THAT ONE WITH 00020000.  THE TEST THEN USES A VIRTUAL ADDRESS
3497                                  ;*      TO SELECT THAT MAP REGISTER AND ADD 17776, SO THAT IT SHOULD
3498                                  ;*      REFERENCE ADDRESS 00037776.  A REFERENCE IS MADE THROUGH EACH
3499                                  ;*      OF THE REGISTERS AND ANY THAT FETCH THE CORRECT DATA ARE CHECKED
3500                                  ;*      TO SEE THAT IT WAS THE MAP REGISTER UNDER TEST.  IF NOT BOTH THE
3501                                  ;*      MAP REGISTER UNDER TEST AND THE DUALED REGISTER ARE REPORTED.
3502                                  ;*
3503                                  ;;**************************************************************
3504  014500                          TST25:
3505  014500  000004                          SCOPE
```

```
3506  014502  012737  014674  001324          MOV     #TST26,NXTTST   ;SAVE STARTING ADDRESS OF NEXT TEST
3507                                                                    ;FOR ESCAPE ON PARITY ERRORS
3508  014510  012737  000144  001204          MOV     #144,$TIMES     ;DO 144 ITERATIONS
3509  014516  004737  005054          20$:    JSR     PC,CLRMAP       ;CLEAR ALL MAP REGISTERS
3510  014522  005037  001170                  CLR     $TMP0           ;USED AS FLAG IN TEST
3511  014526  012703  100000                  MOV     #100000,R3      ;SELECT P.A.R. 4 OFFSET OF ZERO
3512  014532  012737  014564  001110          MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
3513  014540  013702  001244                  MOV     LREGL,R2        ;PUT ADDRESS OF LOWEST USABLE MAP REG IN R2
3514  014544  013700  001240                  MOV     LOWEST,R0       ;MAP REGISTER UNDER TEST IN R0
3515  014550  013701  001240          1$:     MOV     LOWEST,R1       ;MAP REGISTER USED IN CURRENT REFERENCE
3516  014554  012712  040000                  MOV     #40000,(R2)     ;LOAD MAP REG UNDER TEST WITH 8K BASE
3517  014560  010237  040000                  MOV     R2,@#40000      ;LOAD TEST LOCATION WITH THE ADDRESS
3518                                                                    ;OF THE MAP REGISTER UNDER TEST
3519  014564  010137  172350          2$:     MOV     R1,KIPAR4       ;LOAD PAR 4 WITH NEXT MAP REG UNIBUS ADDR
3520  014570  011304                  3$:     MOV     (R3),R4         ;READ THROUGH THE MAP
3521  014572  020402                          CMP     R4,R2           ;SEE IF CORRECT DATA WAS FETCHED
3522  014574  001010                          BNE     4$              ;BRANCH IF NO MATCH
3523  014576  020001                          CMP     R0,R1           ;SEE IF MAP REGS ARE THE SAME
3524  014600  001403                          BEQ     3$              ;BRANCH IF CORRECT MAP REG WAS USED
3525  014602  004737  005276                  JSR     PC,DUALADR      ;LOG AND REPORT ALL ERRORS
3526  014606  000403                          BR      4$              ;SKIP NEXT INSTRUCTION
3527  014610  012737  000001  001170  3$:     MOV     #1,$TMP0        ;SET FLAG WHEN ADDRS MATCH
3528  014616  062701  000200          4$:     ADD     #200,R1         ;TRY NEXT MAP REG
3529  014622  023701  001242                  CMP     HIGEST,R1       ;SEE IF ALL HAVE BEEN TRIED
3530  014626  103356                          BHIS    2$              ;BRANCH IF STILL MORE TO TRY
3531  014630  005737  001170                  TST     $TMP0           ;SEE THAT THERE WAS A SUCCESSFUL MATCH
3532  014634  001001                          BNE     5$              ;BRANCH IF THERE WAS
3533  014636  104036                          ERROR   36              ;DID NOT MATCH MAP REG ADDRESS
3534  014640  005037  001170          5$:     CLR     $TMP0           ;CLEAR FLAG FOR NEXT REG
3535  014644  005012                          CLR     (R2)            ;CLEAR MAP REG JUST TESTED
3536  014646  062702  000004                  ADD     #4,R2           ;POINT TO NEXT MAP REG TO LOAD
3537  014652  062700  000200                  ADD     #200,R0         ;POINT TO NEXT MAP REG UNDER TEST
3538  014656  023700  001242                  CMP     HIGEST,R0       ;SEE IF ALL MAP REGS HAVE BEEN TESTED
3539  014662  103332                          BHIS    1$              ;BRANCH IF STILL MORE TO TEST
3540  014664  005737  001254                  TST     ERRCNT          ;SEE IF THERE WERE ANY ERRORS
3541  014670  001401                          BEQ     TST26           ;;BRANCH TO NEXT TEST IF NO ERRORS
3542  014672  104013                          ERROR   13              ;ERROR TYPE OUT ITEM 13
3543
3544
3545
3546
3547                                  ;;*****************************************************************
3548                                  ;*TEST 26       LOAD LOCATIONS 40000 - 137776 WITH THEIR ADDRESSES
3549                                  ;*
3550                                  ;*      THIS TEST IS USED TO LOAD MAIN MEMORY FROM ADDRESS 00040000 TO
3551                                  ;*      ADDRESS 00137776 WITH ITS OWN ADDRESS.  IT THEN CHECKS THAT
3552                                  ;*      MEMORY OVER THE UNIBUS AND LOGS ANG REPORTS ANY ERRORS THAT
3553                                  ;*      IT FINDS.
3554                                  ;*
3555                                  ;;*****************************************************************
3556  014674                          TST26:
3557  014674  000004                          SCOPE
3558  014676  012737  015154  001324          MOV     #TST27,NXTTST   ;SAVE STARTING ADDRESS OF NEXT TEST
3559                                                                    ;FOR ESCAPE ON PARITY ERRORS
3560  014704  042737  000040  172516          BIC     #BIT5,MMR3      ;TURN OFF MAP RELOCATION
3561  014712  012737  000400  172350          MOV     #400,KIPAR4     ;MAP PAGE 4 TO 8K
```

```
3562  014720  012700  040000              MOV     #40000,R0       ;STARTING ADDRESS FOR DATA PATTERN
3563  014724  012701  100000        1$:   MOV     #100000,R1      ;VIRTUAL ADDRESS
3564  014730  012702  010000              MOV     #^D4096,R2      ;LOAD 4096 LOCATIONS AT A TIME
3565  014734  010021              2$:   MOV     R0,(R1)+        ;LOAD PHY. ADDR. INTO EACH MEMORY LOC.
3566  014736  062700  000002              ADD     #2,R0           ;POINT TO NEXT PHYSICAL ADDRESS
3567  014742  077204                      SOB     R2,2$           ;BRANCH IF 4K OF MEMORY NOT LOADED
3568  014744  062737  000200  172350      ADD     #200,KIPAR4     ;POINT TO NEXT 4K BANK OF MEMORY
3569  014752  022737  001400  172350      CMP     #1400,KIPAR4    ;SEE IF 24K IS LOADED
3570  014760  101361                      BHI     1$              ;BRANCH IF MORE MEMORY TO LOAD
3571                          ;*
3572                          ;*      MEMORY FROM 8K - 24K IS NOW LOADED WITH ITS OWN ADDRESS
3573                          ;*
3574  014762  012737  014770  001106      MOV     #20$,$LPADR     ;SET LOOP ADDRESS TO 20$
3575  014770  012737  015026  001110  20$: MOV     #4$,$LPERR      ;SET LOOP ON ERROR POINTER TO 4$
3576  014776  022737  171400  172354      CMP     #171400,KIPAR6  ;DID I USE ANY MAP REGISTER
3577                                                              ;BELOW REGISTER 6 (UB. ADDR 140000)
3578  015004  101463                      BLOS    TST27           ;;BRANCH TO NEXT TEST IF NOT
3579  015006  013700  172354              MOV     KIPAR6,R0       ;LOAD PAR6 INTO R0 TO GET
3580                                                              ;THE STARTING DATA PATTERN
3581  015012  072027  000006              ASH     #6,R0           ;R0 NOW HOLDS THE STARTING DATA PATTERN
3582  015016  012701  140000        3$:   MOV     #140000,R1      ;STARTING VIRTUAL ADDRESS
3583  015022  012702  010000              MOV     #^D4096,R2      ;PREPARE TO READ 4K AT A TIME
3584  015026  000240                4$:   NOP                     ;THIS IS A SYNC POINT FOR SCOPING
3585  015030  011103                      MOV     (R1),R3         ;READ MAIN MEMORY THRU UNIBUS
3586  015032  020003                      CMP     R0,R3           ;SEE IF THE ADDRESSES MATCH
3587  015034  001015                      BNE     6$              ;BRANCH IF ERROR
3588  015036  062701  000002        5$:   ADD     #2,R1           ;CHANGE VIRTUAL ADDRESS
3589  015042  062700  000002              ADD     #2,R0           ;CHANGE PHYSICAL ADDRESS
3590  015046  077211                      SOB     R2,4$           ;BRANCH IF 4K OF MEMORY NOT READ
3591  015050  062737  000200  172354      ADD     #200,KIPAR6     ;POINT TO NEXT BANK OF 4K THRU UNIBUS
3592  015056  022737  171400  172354      CMP     #171400,KIPAR6  ;SEE IF THIS POINTS TO 24K PLUS 2
3593  015064  101354                      BHI     3$              ;BRANCH IF 24K OF MEMORY NOT CHECKED
3594  015066  000423                      BR      10$             ;TEST FINISHED, BRANCH TO EXIT
3595
3596  015070                        6$:
3597  015070  005100                      COM     R0              ;GET R0 READY FOR AND
3598  015072  040037  001224              BIC     R0,ADRAND               ;PERFORM LOGICAL AND
3599  015076  005100                      COM     R0              ;PUT R0 BACK AS IT WAS
3600  015100  005103                      COM     R3              ;GET R3 READY FOR AND
3601  015102  040337  001230              BIC     R3,DATAND               ;PERFORM LOGICAL AND
3602  015106  005103                      COM     R3              ;PUT R3 BACK AS IT WAS
3603  015110  050037  001226              BIS     R0,ADDROR       ;LOGICAL OR OF PHYSICAL ADDRESS
3604  015114  050337  001232              BIS     R3,DATAOR       ;LOGICAL OR OF ADDRESS FETCHED
3605  015120  005737  001254              TST     ERRCNT          ;IS THIS THE FIRST ERROR HERE?
3606  015124  001002                      BNE     7$              ;BRANCH IF NOT FIRST ERROR
3607  015126  104206                      ERROR   206             ;DIDN'T READ ADDRESSES RIGHT FROM UNIBUS
3608  015130  000742                      BR      5$              ;CONTINUE TESTING
3609  015132  104306                7$:   ERROR   306             ;REPORT MORE DATA FROM UNIBUS
3610  015134  000740                      BR      5$              ;CONTINUE WITH TEST
3611  015136  005737  001254        10$:  TST     ERRCNT          ;WERE THERE ANY ERRORS ON THIS TEST?
3612  015142  001401                      BEQ     19$             ;BRANCH IF NO ERRORS ON THIS TEST
3613  015144  104037                      ERROR   37              ;SUMMARY OF UNIBUS ADDRESS FAILURES
3614  015146  012737  014770  001110  19$: MOV     #20$,$LPERR     ;SET LOOP POINTER TO 20$
3615
3616
3617                          ;;*****************************************************************
```

```
                                        ;*TEST 27       MAIN MEMORY TIMEOUT THROUGH MAP
                                        ;*
                                        ;*      THIS TEST GENERATES A TIME OUT THROUGH THE UNIBUS MAP BY TRYING
                                        ;*    . TO REFERENCE ADDRESS 17000000 IN MAIN MEMORY.  IT USES THE LOWEST
                                        ;*      USEABLE MAP REGISTER, WHICH IN THE DEFAULT CASE IS MAP REGISTER
                                        ;*      ZERO.
                                        ;*
                                        ;;*****************************************************************
015154                                  TST27:
015154  000004                                SCOPE
015156  012737  015266  001324                MOV     #TST30,NXTTST   ;SAVE STARTING ADDRESS OF NEXT TEST
                                                                        ;FOR ESCAPE ON PARITY ERRORS
015164  052737  000040  172516          20$:  BIS     #BIT5,MMR3      ;TURN MAP RELOCATION BACK ON
015172  012737  000020  001264                MOV     #TIMOUT,CPUEXP  ;EXPECTING TIMEOUT IN THIS TEST
015200  013737  001240  172350                MOV     LOWEST,KIPAR4   ;LOAD PAR 4 WITH LOWEST USABLE MAP REG
015206  012777  000074  164032                MOV     #74,@LREGU      ;LOAD UPPER 6 BITS OF LOWEST MAP REG
015214  012777  000000  164022                MOV     #000000,@LREGL  ;LOAD LOWER 16 BITS OF LOWEST MAP REG
015222  012737  015230  001110                MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
015230  005037  001266          1$:   CLR     PCPUER          ;CPU ERROR REG LOCATION
015234  000240                                NOP                     ;THIS IS A SYNC POINT FOR SCOPING
015236  013703  100000                MOV     @#100000,R3     ;TRY TO READ THRU PAGE 4
                                                                        ;THIS REFERENCE WILL GO OUT ON THE
                                                                        ;UNIBUS TO SELECT THE LOWEST USEABLE
                                                                        ;MAP REGISTER (DEFAULT MAP REG. 0).
                                                                        ;PHYSICAL ADDRESS 17700000 IS THEN
                                                                        ;GENERATED, WHICH SHOULD TIME OUT SINCE
                                                                        ;IT IS THE FIRST NON-EXISTANT LOCATION.
015242  022737  000020  001266                CMP     #TIMOUT,PCPUER  ;THE UNIBUS SHOULD HAVE TIMED OUT
015250  001401                                BEQ     10$             ;BRANCH IF CONDITION WAS CORRECT
015252  104040                                ERROR   40              ;UNIBUS DID NOT TIME OUT
015254  012737  015164  001110          10$:  MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
015262  005037  001264                CLR     CPUEXP          ;NO CPU TRAPS EXPECTED FOR AWHILE

                                        ;;*****************************************************************
                                        ;*TEST 30       RELOCATION TEST USING LOWEST USABLE MAPPING REG
                                        ;*
                                        ;*      THIS TEST CHECKS OUT THE FULL ADDITION PROPERTIES OF THE UNIBUS
                                        ;*      MAP A.L.U..  IN THE DEFAULT CASE IT USES MAP REGISTER ZERO BUT
                                        ;*      IF THE MAP JUMPERS HAVE BEEN ALTERED TO DE-SELECT SOME MAP REGISTERS
                                        ;*      THIS TEST WILL USE THE LOWEST USEABLE MAP REGISTER.
                                        ;*      IF AN ERROR OCCURS THE TEST WILL REPORT THE PHYSICAL ADDRESS
                                        ;*      THAT WAS DESIRED, AND THE DATA AT THE ADDRESS THAT WAS REFERENCED.
                                        ;*
                                        ;;*****************************************************************
015266                                  TST30:
015266  000004                                SCOPE
015270  012737  016174  001324                MOV     #TST31,NXTTST   ;SAVE STARTING ADDRESS OF NEXT TEST
                                                                        ;FOR ESCAPE ON PARITY ERRORS
                                        ;;
                                        ;*THE RELOCATION HERE USES A BASE OF 00060000 AND AN
                                        ;*OFFSET OF 00000 TO PRODUCE AN ADDRESS OF 00060000
                                        ;;
015276  012737  015330  001110          100$:  MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
015304  005077  163736                CLR     @LREGU          ;CLEAR UPPER BITS OF MAPPING REG
015310  012777  060000  163726                MOV     #060000,@LREGL  ;LOAD LOWER BITS OF MAPPING REG
```

```
3674  015316  013737  001240  172354          MOV     LOWEST,@#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
3675  015324  012700  140000                  MOV     #140000,R0      ;SELECT PAR6, OFFSET IS 00000
3676  015330  000240                  1$:     NOP                     ;THIS IS A SYNC POINT FOR SCOPING
3677  015332  011001                          MOV     (R0),R1         ;READ LOCATION 060000 THRU THE UNIBUS
3678  015334  012702  060000                  MOV     #060000,R2      ;THE EXPECTED PHYSICAL ADDRESS IS 060000
3679  015340  020102                          CMP     R1,R2           ;SEE IF THE MAP'S FETCH WAS CORRECT
3680  015342  001401                          BEQ     2$              ;BRANCH IF FETCHED DATA MATCHES ADDRESS
3681  015344  104041                          ERROR   41              ;FAULTY RELOCATION BY UNIBUS MAP
3682                                  ;;
3683                                  ;*THE RELOCATION HERE USES A BASE OF 00052524 AND AN
3684                                  ;*OFFSET OF 05252 TO PRODUCE AN ADDRESS OF 00057776
3685                                  ;;
3686  015346  012737  015400  001110  2$:     MOV     #3$,$LPERR      ;SET LOOP ON ERROR POINTER TO 3$
3687  015354  005077  163666                  CLR     @LREGU          ;CLEAR UPPER BITS OF MAPPING REG
3688  015360  012777  052524  163656          MOV     #052524,@LREGL  ;LOAD LOWER BITS OF MAPPING REG
3689  015366  013737  001240  172354          MOV     LOWEST,@#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
3690  015374  012700  145252                  MOV     #145252,R0      ;SELECT PAR6, OFFSET IS 05252
3691  015400  000240                  3$:     NOP                     ;THIS IS A SYNC POINT FOR SCOPING
3692  015402  011001                          MOV     (R0),R1         ;READ LOCATION 057776 THRU THE UNIBUS
3693  015404  012702  057776                  MOV     #057776,R2      ;THE EXPECTED PHYSICAL ADDRESS IS 057776
3694  015410  020102                          CMP     R1,R2           ;SEE IF THE MAP'S FETCH WAS CORRECT
3695  015412  001401                          BEQ     4$              ;BRANCH IF FETCHED DATA MATCHES ADDRESS
3696  015414  104041                          ERROR   41              ;FAULTY RELOCATION BY UNIBUS MAP
3697                                  ;;
3698                                  ;*THE RELOCATION HERE USES A BASE OF 00045252 AND AN
3699                                  ;*OFFSET OF 12524 TO PRODUCE AN ADDRESS OF 00057776
3700                                  ;;
3701  015416  012737  015450  001110  4$:     MOV     #5$,$LPERR      ;SET LOOP ON ERROR POINTER TO 5$
3702  015424  005077  163616                  CLR     @LREGU          ;CLEAR UPPER BITS OF MAPPING REG
3703  015430  012777  045252  163606          MOV     #045252,@LREGL  ;LOAD LOWER BITS OF MAPPING REG
3704  015436  013737  001240  172354          MOV     LOWEST,@#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
3705  015444  012700  152524                  MOV     #152524,R0      ;SELECT PAR6, OFFSET IS 12524
3706  015450  000240                  5$:     NOP                     ;THIS IS A SYNC POINT FOR SCOPING
3707  015452  011001                          MOV     (R0),R1         ;READ LOCATION 057776 THRU THE UNIBUS
3708  015454  012702  057776                  MOV     #057776,R2      ;THE EXPECTED PHYSICAL ADDRESS IS 057776
3709  015460  020102                          CMP     R1,R2           ;SEE IF THE MAP'S FETCH WAS CORRECT
3710  015462  001401                          BEQ     6$              ;BRANCH IF FETCHED DATA MATCHES ADDRESS
3711  015464  104041                          ERROR   41              ;FAULTY RELOCATION BY UNIBUS MAP
3712                                  ;;
3713                                  ;*THE RELOCATION HERE USES A BASE OF 00050420 AND AN
3714                                  ;*OFFSET OF 10420 TO PRODUCE AN ADDRESS OF 00061040
3715                                  ;;
3716  015466  012737  015520  001110  6$:     MOV     #7$,$LPERR      ;SET LOOP ON ERROR POINTER TO 7$
3717  015474  005077  163546                  CLR     @LREGU          ;CLEAR UPPER BITS OF MAPPING REG
3718  015500  012777  050420  163536          MOV     #050420,@LREGL  ;LOAD LOWER BITS OF MAPPING REG
3719  015506  013737  001240  172354          MOV     LOWEST,@#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
3720  015514  012700  150420                  MOV     #150420,R0      ;SELECT PAR6, OFFSET IS 10420
3721  015520  000240                  7$:     NOP                     ;THIS IS A SYNC POINT FOR SCOPING
3722  015522  011001                          MOV     (R0),R1         ;READ LOCATION 061040 THRU THE UNIBUS
3723  015524  012702  061040                  MOV     #061040,R2      ;THE EXPECTED PHYSICAL ADDRESS IS 061040
3724  015530  020102                          CMP     R1,R2           ;SEE IF THE MAP'S FETCH WAS CORRECT
3725  015532  001401                          BEQ     8$              ;BRANCH IF FETCHED DATA MATCHES ADDRESS
3726  015534  104041                          ERROR   41              ;FAULTY RELOCATION BY UNIBUS MAP
3727                                  ;;
3728                                  ;*THE RELOCATION HERE USES A BASE OF 00054630 AND AN
3729                                  ;*OFFSET OF 04210 TO PRODUCE AN ADDRESS OF 00061040
```

```
3730                                            ;;
3731  015536  012737  015570  001110  8$:       MOV     #9$,$LPERR      ;SET LOOP ON ERROR POINTER TO 9$
3732  015544  005077  163476                    CLR     @LREGU          ;CLEAR UPPER BITS OF MAPPING REG
3733  015550  012777  054630  163466            MOV     #054630,@LREGL  ;LOAD LOWER BITS OF MAPPING REG
3734  015556  013737  001240  172354            MOV     LOWEST,@#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
3735  015564  012700  144210                    MOV     #144210,R0      ;SELECT PAR6, OFFSET IS 04210
3736  015570  000240                  9$:       NOP                     ;THIS IS A SYNC POINT FOR SCOPING
3737  015572  011001                            MOV     (R0),R1         ;READ LOCATION 061040 THRU THE UNIBUS
3738  015574  012702  061040                    MOV     #061040,R2      ;THE EXPECTED PHYSICAL ADDRESS IS 061040
3739  015600  020102                            CMP     R1,R2           ;SEE IF THE MAP'S FETCH WAS CORRECT
3740  015602  001401                            BEQ     10$             ;BRANCH IF FETCHED DATA MATCHES ADDRESS
3741  015604  104041                            ERROR   41              ;FAULTY RELOCATION BY UNIBUS MAP
3742                                            ;;
3743                                            ;*THE RELOCATION HERE USES A BASE OF 00044210 AND AN
3744                                            ;*OFFSET OF 14630 TO PRODUCE AN ADDRESS OF 00061040
3745                                            ;;
3746  015606  012737  015640  001110  10$:      MOV     #11$,$LPERR     ;SET LOOP ON ERROR POINTER TO 11$
3747  015614  005077  163426                    CLR     @LREGU          ;CLEAR UPPER BITS OF MAPPING REG
3748  015620  012777  044210  163416            MOV     #044210,@LREGL  ;LOAD LOWER BITS OF MAPPING REG
3749  015626  013737  001240  172354            MOV     LOWEST,@#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
3750  015634  012700  154630                    MOV     #154630,R0      ;SELECT PAR6, OFFSET IS 14630
3751  015640  000240                  11$:      NOP                     ;THIS IS A SYNC POINT FOR SCOPING
3752  015642  011001                            MOV     (R0),R1         ;READ LOCATION 061040 THRU THE UNIBUS
3753  015644  012702  061040                    MOV     #061040,R2      ;THE EXPECTED PHYSICAL ADDRESS IS 061040
3754  015650  020102                            CMP     R1,R2           ;SEE IF THE MAP'S FETCH WAS CORRECT
3755  015652  001401                            BEQ     12$             ;BRANCH IF FETCHED DATA MATCHES ADDRESS
3756  015654  104041                            ERROR   41              ;FAULTY RELOCATION BY UNIBUS MAP
3757                                            ;;
3758                                            ;*THE RELOCATION HERE USES A BASE OF 00056734 AND AN
3759                                            ;*OFFSET OF 02104 TO PRODUCE AN ADDRESS OF 00061040
3760                                            ;;
3761  015656  012737  015710  001110  12$:      MOV     #13$,$LPERR     ;SET LOOP ON ERROR POINTER TO 13$
3762  015664  005077  163356                    CLR     @LREGU          ;CLEAR UPPER BITS OF MAPPING REG
3763  015670  012777  056734  163346            MOV     #056734,@LREGL  ;LOAD LOWER BITS OF MAPPING REG
3764  015676  013737  001240  172354            MOV     LOWEST,@#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
3765  015704  012700  142104                    MOV     #142104,R0      ;SELECT PAR6, OFFSET IS 02104
3766  015710  000240                  13$:      NOP                     ;THIS IS A SYNC POINT FOR SCOPING
3767  015712  011001                            MOV     (R0),R1         ;READ LOCATION 061040 THRU THE UNIBUS
3768  015714  012702  061040                    MOV     #061040,R2      ;THE EXPECTED PHYSICAL ADDRESS IS 061040
3769  015720  020102                            CMP     R1,R2           ;SEE IF THE MAP'S FETCH WAS CORRECT
3770  015722  001401                            BEQ     14$             ;BRANCH IF FETCHED DATA MATCHES ADDRESS
3771  015724  104041                            ERROR   41              ;FAULTY RELOCATION BY UNIBUS MAP
3772                                            ;;
3773                                            ;*THE RELOCATION HERE USES A BASE OF 00042104 AND AN
3774                                            ;*OFFSET OF 16734 TO PRODUCE AN ADDRESS OF 00061040
3775                                            ;;
3776  015726  012737  015760  001110  14$:      MOV     #15$,$LPERR     ;SET LOOP ON ERROR POINTER TO 15$
3777  015734  005077  163306                    CLR     @LREGU          ;CLEAR UPPER BITS OF MAPPING REG
3778  015740  012777  042104  163276            MOV     #042104,@LREGL  ;LOAD LOWER BITS OF MAPPING REG
3779  015746  013737  001240  172354            MOV     LOWEST,@#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
3780  015754  012700  156734                    MOV     #156734,R0      ;SELECT PAR6, OFFSET IS 16734
3781  015760  000240                  15$:      NOP                     ;THIS IS A SYNC POINT FOR SCOPING
3782  015762  011001                            MOV     (R0),R1         ;READ LOCATION 061040 THRU THE UNIBUS
3783  015764  012702  061040                    MOV     #061040,R2      ;THE EXPECTED PHYSICAL ADDRESS IS 061040
3784  015770  020102                            CMP     R1,R2           ;SEE IF THE MAP'S FETCH WAS CORRECT
3785  015772  001401                            BEQ     16$             ;BRANCH IF FETCHED DATA MATCHES ADDRESS
```

```
                                        ERROR   41              ;FAULTY RELOCATION BY UNIBUS MAP
                                ;;
                                ;*THE RELOCATION HERE USES A BASE OF 00057776 AND AN
                                ;*OFFSET OF 01042 TO PRODUCE AN ADDRESS OF 00061040
                                ;;
016774  012737  016030  001110  16$:    MOV     #17$,$LPERR     ;SET LOOP ON ERROR POINTER TO 17$
016004  005077  163236          16$:    CLR     @LREGU          ;CLEAR UPPER BITS OF MAPPING REG
016010  012777  057776  163226          MOV     #057776,@LREGL  ;LOAD LOWER BITS OF MAPPING REG
016016  013737  001240  172354          MOV     LOWEST,@#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
016024  012700  141042                  MOV     #141042,R0      ;SELECT PAR6, OFFSET IS 01042
016030  000240                  17$:    NOP                     ;THIS IS A SYNC POINT FOR SCOPING
016032  011001                          MOV     (R0),R1         ;READ LOCATION 061040 THRU THE UNIBUS
016034  012702  061040                  MOV     #061040,R2      ;THE EXPECTED PHYSICAL ADDRESS IS 061040
016040  020102                          CMP     R1,R2           ;SEE IF THE MAP'S FETCH WAS CORRECT
016042  001401                          BEQ     18$             ;BRANCH IF FETCHED DATA MATCHES ADDRESS
016044  104041                          ERROR   41              ;FAULTY RELOCATION BY UNIBUS MAP
                                ;;
                                ;*THE RELOCATION HERE USES A BASE OF 00041042 AND AN
                                ;*OFFSET OF 17776 TO PRODUCE AN ADDRESS OF 00061040
                                ;;
016046  012737  016100  001110  18$:    MOV     #19$,$LPERR     ;SET LOOP ON ERROR POINTER TO 19$
016054  005077  163166          18$:    CLR     @LREGU          ;CLEAR UPPER BITS OF MAPPING REG
016060  012777  041042  163156          MOV     #041042,@LREGL  ;LOAD LOWER BITS OF MAPPING REG
016066  013737  001240  172354          MOV     LOWEST,@#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
016074  012700  157776                  MOV     #157776,R0      ;SELECT PAR6, OFFSET IS 17776
016100  000240                  19$:    NOP                     ;THIS IS A SYNC POINT FOR SCOPING
016102  011001                          MOV     (R0),R1         ;READ LOCATION 061040 THRU THE UNIBUS
016104  012702  061040                  MOV     #061040,R2      ;THE EXPECTED PHYSICAL ADDRESS IS 061040
016110  020102                          CMP     R1,R2           ;SEE IF THE MAP'S FETCH WAS CORRECT
016112  001401                          BEQ     20$             ;BRANCH IF FETCHED DATA MATCHES ADDRESS
016114  104041                          ERROR   41              ;FAULTY RELOCATION BY UNIBUS MAP
                                ;;
                                ;*THE RELOCATION HERE USES A BASE OF 00057776 AND AN
                                ;*OFFSET OF 00002 TO PRODUCE AN ADDRESS OF 00060000
                                ;;
016116  012737  016150  001110  20$:    MOV     #21$,$LPERR     ;SET LOOP ON ERROR POINTER TO 21$
016124  005077  163116          20$:    CLR     @LREGU          ;CLEAR UPPER BITS OF MAPPING REG
016130  012777  057776  163106          MOV     #057776,@LREGL  ;LOAD LOWER BITS OF MAPPING REG
016136  013737  001240  172354          MOV     LOWEST,@#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
016144  012700  140002                  MOV     #140002,R0      ;SELECT PAR6, OFFSET IS 00002
016150  000240                  21$:    NOP                     ;THIS IS A SYNC POINT FOR SCOPING
016152  011001                          MOV     (R0),R1         ;READ LOCATION 060000 THRU THE UNIBUS
016154  012702  060000                  MOV     #060000,R2      ;THE EXPECTED PHYSICAL ADDRESS IS 060000
016160  020102                          CMP     R1,R2           ;SEE IF THE MAP'S FETCH WAS CORRECT
016162  001401                          BEQ     22$             ;BRANCH IF FETCHED DATA MATCHES ADDRESS
016164  104041                          ERROR   41              ;FAULTY RELOCATION BY UNIBUS MAP
016166  012737  015276  001110  22$:    MOV     #100$,$LPERR    ;SET LOOP POINTER TO START OF TEST

                                ;;**************************************************************
                                ;*TEST 31        TEST CARRY PROPAGATION OF MAP'S RELOCATION ADDER
                                ;*
                                ;*      EVERY ADDRESS OF THE FORM XXXX0000 IS GENERATED HERE STARTING
                                ;*      WITH 00030000 UP TO 17000000.  THAT IS, THE FIRST OF EVERY 2K
                                ;*      WORDS IS ADDRESSED, TO INSURE THAT THE ADDER IN THE MAP IS
                                ;*      WORKING PROPERLY AND, THE SYSTEM SIZE JUMPERS ARE ACTUALLY SET
```

```
                                        ;*      FOR THE TOP OF MAIN MEMORY.
                                        ;*
                                        ;*************************************************************
016174                                  TST31:
016174  000004                                  SCOPE
016176  012737  016464  001324                  MOV     #TST32,NXTTST   ;SAVE STARTING ADDRESS OF NEXT TEST
                                                                        ;FOR ESCAPE ON PARITY ERRORS
016204  012737  000012  001204                  MOV     #12,$TIMES      ;DO 12 ITERATIONS
                                                .EQUIV  BIT5,NEXMEM     ;BIT05 IS NON-EXISTANT MEMORY BIT IN
                                                                        ;THE CPU ERROR REGISTER
016212  012737  177777  016346  20$:   MOV     #-1,4$          ;INITIALIZE FLAG AS NEGATIVE ONE
016220  005077  163022                  CLR     @LREGU          ;CLEAR UPPER 6 BITS OF MAP REG
016224  012777  020000  163012          MOV     #20000,@LREGL   ;LOAD 4K BASE INTO MAP REGISTER
016232  012701  100100                  MOV     #100100,R1      ;LOAD BITS TO SELECT PAR 4, OFFSET 100
016236  012700  150000                  MOV     #150000,R0      ;LOAD BITS TO SELECT PAR 6, OFFSET 2K
016242  012737  000277  172350          MOV     #277,KIPAR4     ;START WITH PHYSICAL 6K
016250  013737  001240  172354          MOV     LOWEST,KIPAR6   ;LOAD PAR 6 WITH MAP REG'S ADDR
016256  012737  016316  001110          MOV     #10$,$LPERR     ;SET LOOP ON ERROR POINTER TO 10$
016264  012737  000020  001264  3$:     MOV     #20,CPUEXP      ;EXPECTING A UNIBUS TIME OUT DURING TEST
016272  005037  001266                  CLR     PCPUER          ;CLEAR TIME OUT FLAG
016276  013710  001326                  MOV     DATA,(R0)       ;THIS LOAD WILL TIME OUT WHEN YOU
                                                                ;HAVE REACHED THE TOP OF MEMORY
                                                                ;IT SELECTS PAR 6 WHICH WILL PUT ADDR
                                                                ;<XXX XX1>0000 ON THE UNIBUS.
                                                                ;THE X'S WILL SELECT THE LOWEST USEABLE
                                                                ;MAPPING REGISTER.  THE DEFAULT CASE IS
                                                                ;010000,  SELECTING MAPPING REGISTER 0.
016302  005737  001266                  TST     PCPUER          ;SEE IF THERE WAS MAIN MEMORY
016306  001016                          BNE     1$              ;BRANCH IF NO MAIN MEMORY FROM UNIBUS
016310  012737  000040  001264          MOV     #NEXMEM,CPUEXP  ;POSSIBLE CACHE NON-EXISTANT MEMORY
016316  011103                  10$:    MOV     (R1),R3         ;READ TEST LOCATION VIA FASTBUS
016320  022737  000040  001266          CMP     #NEXMEM,PCPUER  ;WAS THIS CACHE NON-EXISTANT MEMORY
016326  001414                          BEQ     2$              ;BRANCH IF NON-EXSITANT MEMORY
016330  000240                          NOP                     ;THIS IS A SYNC POINT FOR SCOPING
016332  011002                          MOV     (R0),R2         ;READ TEST LOCATION VIA UNIBUS MAP
016334  020203                          CMP     R2,R3           ;COMPARE TEST DATA R2=MAP DATA
                                                                ;R3=FASTBUS DATA
016336  001410                          BEQ     2$              ;BRANCH IF IT WAS THE SAME
016340  104042                          ERROR   42              ;CARRY PROPAGATION FAILURE IN MAP
016342  000406                          BR      2$              ;BRANCH TO UPDATE ROUTINE
016344  005227                  1$:     INC     (PC)+           ;INCREMENT ONE TIME ENTRANCE FLAG
016346  177777                  4$:     .WORD   -1              ;USE NEGATIVE ONE FOR FLAG
016350  001003                          BNE     2$              ;BRANCH IF YOU'VE BEEN HERE BEFORE
016352  013737  172350  001320          MOV     KIPAR4,RSIZE    ;SAVE UPPER LIMIT OF MEMORY
016360  062737  000100  001326  2$:     ADD     #100,DATA       ;CHANGE PATTERN FOR NEXT LOAD
016366  062737  000100  172350          ADD     #100,KIPAR4     ;ADD 2K TO PAR4
016374  062777  010000  162642          ADD     #10000,@LREGL   ;ADD 2K TO MAP REGISTER
016402  001330                          BNE     3$              ;BRANCH IF MAP REGISTER NOT ZERO
016404  005277  162636                  INC     @LREGU          ;ADD ONE TO UPPER 6 BITS OF MAP REG
016410  022777  000073  162630          CMP     #73,@LREGU      ;SEE IF TOP 128K BLOCK HAS BEEN PASSED
016416  103322                          BHIS    3$              ;BRANCH IF NOT PAST IT
016420  005237  001326                  INC     DATA            ;CHANGE DATA PATTERN FOR NEXT PASS
016424  042737  177700  001326          BIC     #177700,DATA    ;CLEAR UPPER 10 BITS OF DATA PATTERN
016432  052737  000300  001326          BIS     #300,DATA       ;START WITH 3XX IN DATA PATTERN
016440  012737  016212  001110          MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
016446  023737  177760  001320          CMP     @#SIZELO,RSIZE  ;SEE IF SIZE JUMPERS AGREE WITH MEMORY TOP
```

```
                                        BEQ     19$             ;BRANCH IF TOP OF MEMORY AGREES WITH
                                                                ;THE SIZE JUMPERS
                                        ERROR   43              ;TOP OF MEMORY DOESN'T MATCH SIZE JUMPER
                        19$:    CLR     CPUEXP                  ;NO CPU TRAPS EXPECTED FOR AWHILE


                        ;;**********************************************************************
                        ;*TEST 32       PARITY REPORTING THRU THE MAP, MAIN MEMORY EVEN WORD
                        ;*
                        ;*      THIS TEST FORCES A PARITY ERROR IN MAIN MEMORY AT PHYSICAL
                        ;*      ADDRESS 00040000 BY SETTING ONE BIT IN EACH BYTE OF THE
                        ;*      WORD.  THE TEST THEN FORCES THE EVEN WORD PARITY BITS TO
                        ;*      ONES AND READS ADDRESS 00040000 THRU THE UNIBUS MAP.  THEN
                        ;*      IT CHECKS TO SEE THAT THE PARITY ABORT OCCURRED AND THAT THE
                        ;*      CONDITION WAS CORRECT BEFORE GOING TO THE NEXT TEST.
                        ;*
                        ;*      ERRORS ARE REPORTED INLINE IN THE TEST CODE, AND THE PARITY
                        ;*      VECTOR WILL TRAP TO LABEL 10$ AT THE END OF THIS TEST.
                        ;*
                        ;;**********************************************************************
                        TST32:
                                        SCOPE
                                        MOV     #TST33,NXTTST   ;SAVE STARTING ADDRESS OF NEXT TEST
                                                                ;FOR ESCAPE ON PARITY ERRORS
                                        TBITO                   ;TURN OFF T BIT TRAPPING IF ON
                        20$:    MOV     #23404,$TMP4            ;EXPECTED ERROR CONDITION
                                CLR     PPARER                  ;CLEAR MEM ERROR REG STORAGE
                                MOV     #-1,@#MEMERR            ;CLEAR MEMORY ERROR REGISTER
                                MOV     @#CONTRL,$TMP5          ;SAVE CONTROL REG TO RESTORE CACHE
                                MOV     #14,@#CONTRL            ;FORCE MISSES IN CACHE
                                MOV     LOWEST,KIPAR6   ;PUT UNIBUS ADDR OF MAP REG IN PAR6
                                MOV     #40000,@LREGL           ;LOWEST MAP REGISTER POINTS TO 8K
                                CLR     @LREGU                  ;CLEAR UPPER BITS OF MAP REG
                                MOV     #MAINT,R1               ;PUT ADDR OF MAINT REG IN R1
                                MOV     #1$,$LPERR              ;SET LOOP ON ERROR POINTER TO 1$
                                MOV     #140000,R0              ;SELECT KIPAR6, OFFSET 0
                                MOV     #20001,(R0)             ;LOAD ONE BIT IN EACH BYTE OF 40000
                                MOV     #10$,CACHVEC            ;SET PARITY VECTOR TO 10$
                        1$:     MOV     #30000,@#MAINT          ;FORCE EVEN WORD PARITY BITS TO ONE
                                NOP                             ;THIS IS A SYNC POINT FOR SCOPING
                                MOV     (R0),R3                 ;"DATA FETCH" SHOULD CAUSE PARITY ABORT
                                MOV     (R1),R2                 ;SAVE MAINT REG IN CASE NO TRAP
                                CLR     (R1)                    ;CLEAR MAINT REG IN CASE NO TRAP
                                BIS     R0,R0                   ;DUMMY INST WITH P.B.'S ON
                                CMP     $TMP4,PPARER            ;SEE IF ERROR REG HOLDS RIGHT DATA
                                BEQ     2$                      ;BRANCH IF CONDITION WAS CORRECT
                                ERROR   44                      ;PARITY REPORTING BAD
                        2$:     MOV     #-1,MEMERR              ;CLEAR MEMORY ERROR REGISTER
                                MOV     #MEMER,CACHVEC          ;RESTORE PARITY SERVICE ROUTINE
                                MOV     $TMP5,CONTRL            ;RESTORE CACHE TO FORMER CONDITION
                                MOV     #20$,$LPERR             ;SET LOOP POINTER TO START OF TEST
                                BR      TST33                   ;;TEST OVER GO TO NEXT TEST


                        10$:    MOV     @#LOADRS,PLOADR ;SAVE LOWER CACHE ADDR REG
```

```
3954  016674  013737  177742  001272          MOV     @#HIADRS,PHIADR ;SAVE HIGH BITS OF FAILING ADDR
3955  016702  013737  177744  001274          MOV     @#MEMERR,PPARER ;SAVE MEMORY ERROR REGISTER
3956  016710  013737  177746  001276          MOV     @#CONTRL,PCONTR ;SAVE CONTROL REGISTER FOR TYPE OUT
3957  016716  012737  030000  001300          MOV     #30000,PMAINT   ;SAVE DATA IN MAINTENENCE REGISTER
3958  016724  000002                          RTI                     ;RETURN TO TEST AND CHECK PARITY
3959
3960
3961                                  ;;*****************************************************************
3962                                  ;*TEST 33       PARITY REPORTING THRU THE MAP, MAIN MEMORY ODD WORD
3963                                  ;*
3964                                  ;*      THIS TEST FORCES A PARITY ERROR IN MAIN MEMORY AT PHYSICAL
3965                                  ;*      ADDRESS 00040002 BY SETTING ONE BIT IN EACH BYTE OF THE
3966                                  ;*      WORD.  THE TEST THEN FORCES THE ODD WORD PARITY BITS TO
3967                                  ;*      ONES AND READS ADDRESS 00040002 THRU THE UNIBUS MAP.  THEN
3968                                  ;*      IT CHECKS TO SEE THAT THE PARITY ABORT OCCURRED AND THAT THE
3969                                  ;*      CONDITION WAS CORRECT BEFORE GOING TO THE NEXT TEST.
3970                                  ;*
3971                                  ;*      ERRORS ARE REPORTED INLINE IN THE TEST CODE, AND THE PARITY
3972                                  ;*      VECTOR WILL TRAP TO LABEL 10$ AT THE END OF THIS TEST.
3973                                  ;*
3974                                  ;;*****************************************************************
3975  016726                          TST33:
3976  016726  000004                          SCOPE
3977  016730  012737  017200  001324          MOV     #TST34,NXTTST   ;SAVE STARTING ADDRESS OF NEXT TEST
3978                                                                  ;FOR ESCAPE ON PARITY ERRORS
3979  016736  012737  023410  001200  20$:    MOV     #23410,$TMP4    ;LOAD EXPECTED ERROR CONDITION
3980  016744  005037  001274                  CLR     PPARER          ;CLEAR MEM ERROR REG STORAGE
3981  016750  012737  177777  177744          MOV     #-1,@#MEMERR    ;CLEAR MEMORY ERROR REGISTER
3982  016756  013737  177746  001202          MOV     @#CONTRL,$TMP5  ;SAVE CONTROL REG TO RESTORE CACHE
3983  016764  012737  000014  177746          MOV     #14,@#CONTRL    ;FORCE MISSES IN CACHE
3984  016772  013737  001240  172354          MOV     LOWEST,KIPAR6   ;PUT UNIBUS ADDR OF MAP REG IN PAR6
3985  017000  012777  040000  162236          MOV     #40000,@LREGL   ;LOWEST MAP REGISTER POINTS TO 8K
3986  017006  005077  162234                  CLR     @LREGU          ;CLEAR UPPER BITS OF MAP REG
3987  017012  012701  177750                  MOV     #MAINT,R1       ;PUT ADDR OF MAINT REG IN R1
3988  017016  012737  017054  001110          MOV     #1$,$LPERR      ;SET LOOP ON EROR POINTER TO 1$
3989  017024  005037  140000                  CLR     140000          ;CLEAR LOCATION 40000
3990  017030  012700  140002                  MOV     #140002,R0      ;LOAD ADDRESS OF 40002 INTO R0
3991  017034  012737  017054  001110          MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
3992  017042  012710  020001                  MOV     #20001,(R0)     ;LOAD ONE BIT IN EACH BYTE OF 40002
3993  017046  012737  017140  000114          MOV     #10$,CACHVEC    ;SET PARITY VECTOR TO 10$
3994  017054  012737  140000  177750  1$:     MOV     #140000,@#MAINT ;FORCE ODD WORD PARITY BITS TO ONE
3995  017062  000240                          NOP                     ;THIS IS A SYNC POINT FOR SCOPING
3996  017064  011003                          MOV     (R0),R3         ;"DATA FETCH" SHOULD CAUSE PARITY ABORT
3997  017066  011102                          MOV     (R1),R2         ;SAVE MAINT REG IN CASE NO TRAP
3998  017070  005011                          CLR     (R1)            ;CLEAR MAINT REG IN CASE NO TRAP
3999  017072  050000                          BIS     R0,R0           ;DUMMY INST WITH P.B.'S ON
4000  017074  023737  001200  001274          CMP     $TMP4,PPARER    ;SEE IF MEM ERROR REG HOLDS RIGHT DATA
4001  017102  001401                          BEQ     2$              ;BRANCH IF ERROR WAS EXPECTED ONE
4002  017104  104044                          ERROR   44              ;PARITY REPORTING BAD
4003  017106  012737  177777  177744  2$:     MOV     #-1,MEMERR      ;CLEAR MEMORY ERROR REGISTER
4004  017114  012737  005704  000114          MOV     #MEMER,CACHVEC  ;RESTORE PARITY SERVICE ROUTINE
4005  017122  013737  001202  177746          MOV     $TMP5,CONTRL    ;RESTORE CACHE TO FORMER CONDITION
4006  017130  012737  016736  001110          MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
4007  017136  000420                          BR      TST34           ;;TEST IS OVER, GO TO NEXT TEST
4008
4009
```

```
4010  017140  013737  177740  001270  10$:    MOV     @#LOADRS,PLOADR ;SAVE LOWER CACHE ADDR REG
4011  017146  013737  177742  001272          MOV     @#HIADRS,PHIADR ;SAVE HIGH BITS OF FAILING ADDR
4012  017154  013737  177744  001274          MOV     @#MEMERR,PPARER ;SAVE MEMORY ERROR REGISTER
4013  017162  013737  177746  001276          MOV     @#CONTRL,PCONTR ;SAVE CONTROL REGISTER FOR TYPE OUT
4014  017170  012737  140000  001300          MOV     #140000,PMAINT  ;SAVE DATA IN MAINTENENCE REGISTER
4015  017176  000002                          RTI                     ;RETURN TO TEST AND CHECK FARITY
4016
4017
4018                                  ;;*****************************************************************
4019                                  ;*TEST 34      PARITY REPORTING THRU THE MAP, CACHE DATA GROUP 0
4020                                  ;*
4021                                  ;*      THIS TEST FORCES A PARITY ERROR IN CACHE GROUP 0 AT PHYSICAL
4022                                  ;*      ADDRESS 00040000 BY SETTING BIT 15 IN THE WORD.  THE TEST
4023                                  ;*      THEN FORCES GROUP 0 LOW BYTE PARITY BIT TO ZERO AND EVEN .
4024                                  ;*      WORD HIGH BYTE PARITY BIT TO ONE BEFORE READING ADDRESS 00040000
4025                                  ;*      THRU THE UNIBUS MAP.  THEN IT CHECKS TO SEE THAT THE PARITY
4026                                  ;*      ABORT OCCURRED AND THAT THE CONDITION WAS CORRECT BEFORE
4027                                  ;*      GOING TO THE NEXT TEST.
4028                                  ;*
4029                                  ;*      ERRORS ARE REPORTED INLINE IN THE TEST CODE, AND THE PARITY
4030                                  ;*      VECTOR WILL TRAP TO LABEL 10$ AT THE END OF THIS TEST.
4031                                  ;*
4032                                  ;;*****************************************************************
4033  017200                          TST34:
4034  017200  000004                         SCOPE
4035  017202  012737  017466  001324          MOV     #TST35,NXTTST   ;SAVE STARTING ADDRESS OF NEXT TEST
4036                                                                  ;FOR ESCAPE ON PARITY ERRORS
4037  017210  012737  177777  017354  20$:    MOV     #-1,21$         ;RESTORE NEG ONE FOR NEXT ITERATION
4038  017216  032737  000004  177746          BIT     #BIT2,@#CONTRL  ;SEE IF GROUP 0 IS DISABLED
4039  017224  001120                          BNE     TST35           ;;BRANCH IF GROUP 0 IS DISABLED
4040  017226  013737  177746  001202          MOV     CONTRL,$TMP5    ;SAVE CONDITION OF CACHE
4041  017234  012737  023504  001200          MOV     #23504,$TMP4    ;EXPECTED PARITY CONDITION GROUP 0
4042  017242  005037  001274                  CLR     PPARER          ;CLEAR MEM ERROR REG STORAGE
4043  017246  012737  177777  177744          MOV     #-1,@#MEMERR    ;CLEAR MEMORY ERROR REGISTER
4044  017254  013737  001240  172354          MOV     LOWEST,KIPAR6   ;PUT UNIBUS ADDR OF MAP REG IN PAR6
4045  017262  005077  161760                  CLR     @LREGU          ;CLEAR UPPER BITS ON MAP REG
4046  017266  012777  040000  161750          MOV     #40000,@LREGL   ;LOAD LOWER 16 BITS OF MAP REG
4047  017274  012705  177750                  MOV     #MAINT,R5       ;PUT ADDR OF MAINT REG IN R5
4048  017300  012704  140000                  MOV     #140000,R4      ;SELECT PAR 6 BASE 0
4049  017304  012714  100000                  MOV     #BIT15,(R4)     ;MAKE SURE I GET SOFT AND HARD ERRORS
4050                                                                  ;BY LOADING 40000 WITH BIT 15, THIS
4051                                                                  ;CAUSES THE HIGH BYTE P.B. TO BE ON
4052                                                                  ;AND THE LOW BYTE TO BE OFF
4053  017310  012737  000030  177746          MOV     #30,@#CONTRL    ;FORCE SELECTION OF GROUP 0 ON READ
4054  017316  011401                          MOV     (R4),R1         ;PUT (40000) & (40002) IN GROUP 0
4055  017320  005001                          CLR     R1              ;ZERO R1 FOR THE MOMENT
4056  017322  012737  017426  000114          MOV     #10$,CACHVEC    ;SET PARITY VECTOR TO 10$
4057  017330  010137  177750          1$:     MOV     R1,@#MAINT      ;FORCE GPO PARITY BITS TO 0 ON 2ND PASS
4058  017334  000241                          CLC                     ;THIS IS A SYNC POINT FOR SCOPING
4059                                                                  ;THE LOW BYTE HAS ITS P.B. OFF
4060  017336  011401                          MOV     (R4),R1         ;DATA FETCH SHOULD CAUSE PARITY ABORT
4061  017340  011503                          MOV     (R5),R3         ;SAVE MAINT REG IN CASE NO TRAP
4062  017342  105015                          CLRB    (R5)            ;CLEAR MAINT REG IN CASE NO TRAP
4063  017344  006701                          SXT     R1              ;DUMMY INST WITH P.B.'S OFF
4064  017346  012701  020020                  MOV     #020020,R1      ;FORCE GROUP 0 LOW BYTE PARITY BIT
4065                                                                  ;TO 0 AND MAIN MEMORY EVEN WORD HIGH
```

```
                                                        ;BYTE PARITY BIT TO 1
                        INC     (PC)+                   ;INCREMENT NEXT WORD TO ZERO ON FIRST PASS
                21$:    .WORD   -1                      ;
                        BEQ     1$                      ;BRANCH ON FIRST PASS ONLY
                        CMP     $TMP4,PPARER            ;SEE IF ERROR CONDITION MATCHES EXPECTED
                        BEQ     2$                      ;BRANCH IF CORRECT CONDITICN
                        MOV     R3,R2                   ;R2 HOLDS MAINT REG FOR ERROR TYPE OUT
                        ERROR   44                      ;PARITY REPORTING BAD
                2$:     MOV     #-1,MEMERR              ;CLEAR MEMORY ERROR REGISTER
                        MOV     #MEMER,CACHVEC          ;RESTORE PARITY SERVICE ROUTINE
                        MOV     $TMP5,CONTRL            ;RESTORE CACHE TO FORMER CONDITION
                        MOV     #20$,$LPERR             ;SET LOOP POINTER TO START OF TEST
                        BR      TST35                   ;;TEST OVER GO TO NEXT TEST

                10$:    MOV     @#LOADRS,PLOADR         ;SAVE LOWER CACHE ADDR REG
                        MOV     @#HIADRS,PHIADR         ;SAVE HIGH BITS OF FAILING ADDR
                        MOV     @#MEMERR,PPARER         ;SAVE MEMORY ERROR REGISTER
                        MOV     @#CONTRL,PCONTR         ;SAVE CONTROL REGISTER FOR TYPE OUT
                        MOV     #30060,PMAINT           ;SAVE DATA IN MAINTENENCE REGISTER
                        RTI                             ;RETURN TO TEST AND CHECK PARITY

;;*****************************************************************
;*TEST 35        PARITY REPORTING THRU THE MAP, CACHE DATA GROUP 1
;*
;*      THIS TEST FORCES A PARITY ERROR IN CACHE GROUP 1 AT PHYSICAL
;*      ADDRESS 00040000 BY SETTING BIT 15 IN THE WORD.  THE TEST
;*      THEN FORCES GROUP 1 LOW BYTE PARITY BIT TO ZERO AND EVEN
;*      WORD HIGH BYTE PARITY BIT TO ONE BEFORE READING ADDRESS 00040000
;*      THRU THE UNIBUS MAP.  THEN IT CHECKS TO SEE THAT THE PARITY
;*      ABORT OCCURRED AND THAT THE CONDITION WAS CORRECT BEFORE
;*      GOING TO THE NEXT TEST.
;*
;*      ERRORS ARE REPORTED INLINE IN THE TEST CODE, AND THE PARITY
;*      VECTOR WILL TRAP TO LABEL 10$ AT THE END OF THIS TEST.
;*
;*****************************************************************
TST35:
                        SCOPE
                        MOV     #TST36,NXTTST   ;SAVE STARTING ADDRESS OF NEXT TEST
                                                ;FOR ESCAPE ON PARITY ERRORS
                20$:    MOV     #-1,21$         ;RESTORE NEG ONE FOR NEXT ITERATION
                        BIT     #BIT3,CONTRL    ;SEE IF GROUP 1 IS DISABLED
                        BNE     UBMAP           ;;BRANCH TO NEXT TEST IF GROUP 1 DISABLED
                        MOV     CONTRL,$TMP5    ;SAVE CONDITION OF CACHE
                        MOV     #23604,$TMP4    ;EXPECTED PARITY CONDITION GROUP 1
                        CLR     PPARER          ;CLEAR MEM ERROR REG STORAGE
                        MOV     #-1,@#MEMERR    ;CLEAR MEMORY ERROR REGISTER
                        MOV     LOWEST,KIPAR6   ;PUT UNIBUS ADDR OF MAP REG IN PAR6
                        CLR     @LREGU          ;CLEAR UPPER BITS ON MAP REG
                        MOV     #40000,@LREGL   ;LOAD LOWER 16 BITS OF MAP REG
                        MOV     #MAINT,R5       ;PUT ADDR OF MAINT REG IN R5
                        MOV     #140000,R4      ;SELECT PAR 6 BASE 0
                        MOV     #BIT15,(R4)     ;MAKE SURE I GET SOFT AND HARD ERRORS
                                                ;BY LOADING 40000 WITH BIT 15, THIS
```

```
                                                                ;CAUSES THE HIGH BYTE P.B. TO BE ON
                                                                ;AND THE LOW BYTE TO BE OFF
017576  012737  000044  177746          MOV     #44,@#CONTRL    ;FORCE SELECTION OF GROUP 0 ON READ
017604  011401                        MOV     (R4),R1         ;PUT (40000) & (40002) IN GROUP 1
017606  005001                        CLR     R1              ;CLEAR R1 FOR MOMENT
017610  012737  017714  000114        MOV     #10$,CACHVEC    ;SET PARITY VECTOR TO 10$
017616  010137  177750          1$:   MOV     R1,@#MAINT      ;FORCE GP1 PARITY BITS TO 0 ON 2ND PASS
017622  000241                        CLC                     ;THIS IS A SYNC POINT FOR SCOPING
                                                                ;THE LOW BYTE HAS ITS P.B. OFF
017624  011401                        MOV     (R4),R1         ;DATA FETCH SHOULD CAUSE PARITY ABORT
017626  011503                        MOV     (R5),R3         ;SAVE MAINT REG IN CASE NO TRAP
017630  105015                        CLRB    (R5)            ;CLEAR MAINT REG IN CASE NO TRAP
017632  006701                        SXT     R1              ;DUMMY INST WITH P.B.'S OFF
017634  012701  020100                MOV     #020100,R1      ;FORCE GROUP 1 LOW BYTE PARITY BIT
                                                                ;TO 0 AND MAIN MEMORY EVEN WORD HIGH
                                                                ;BYTE PARITY BIT TO 1
017640  005227                        INC     (PC)+           ;INCREMENT NEXT WORD TO 0
017642  177777                21$:    .WORD   -1              ;
017644  001764                        BEQ     1$              ;BRANCH ON FIRST PASS ONLY
017646  023737  001200  001274        CMP     $TMP4,PPARER    ;SEE IF ERROR CONDITION MATCHES EXPECTED
017654  001402                        BEQ     2$              ;BRANCH IF ERROR WAS EXPECTED ONE
017656  010302                        MOV     R3,R2           ;R2 HOLDS MAINT REG FOR ERROR TYPE OUT
017660  104044                        ERROR   44              ;PARITY REPORTING BAD
017662  012737  177777  177744  2$:   MOV     #-1,MEMERR      ;CLEAR MEMORY ERROR REGISTER
017670  012737  005704  000114        MOV     #MEMER,CACHVEC  ;RESTORE PARITY SERVICE ROUTINE
017676  013737  001202  177746        MOV     $TMP5,CONTRL    ;RESTORE CACHE TO FORMER CONDITION
017704  012737  017476  001110        MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
017712  000420                        BR      UBMAP           ;;TEST OVER GO TO NEXT TEST

017714  013737  177740  001270  10$:   MOV     @#LOADRS,PLOADR ;SAVE LOWER CACHE ADDR REG
017722  013737  177742  001272        MOV     @#HIADRS,PHIADR ;SAVE HIGH BITS OF FAILING ADDR
017730  013737  177744  001274        MOV     @#MEMERR,PPARER ;SAVE MEMORY ERROR REGISTER
017736  013737  177746  001276        MOV     @#CONTRL,PCONTR ;SAVE CONTROL REGISTER FOR TYPE OUT
017744  012737  030300  001300        MOV     #30300,PMAINT   ;SAVE DATA IN MAINTENENCE REGISTER
017752  000002                        RTI                     ;RETURN TO TEST AND CHECK PARITY


                        .SBTTL  ************************************************************
                        .SBTTL  THE FOLLOWING TESTS ARE RUN THROUGH THE UNIBUS MAP
                        .SBTTL  ************************************************************

017754  000004          UBMAP:  SCOPE                   ;LOOP ON PREVIOUS TEST
017756  104420                  TBITR                   ;RESTORE T-BIT IF IT WAS ON

                        ;*
                        ;*      THIS CODE SETS UP THE TWO MAP REGISTERS ABOVE THE LOWEST
                        ;*      USEABLE ONE TO POINT TO PHYSICAL MEMORY FROM 0 - 8K.  IN THE
                        ;*      DEFAULT CASE, IN MANUFACTURING AND IF THERE IS NO UNIBUS MEMORY,
                        ;*      THIS WILL BE MAP REGISTERS 1 AND 2.  MAP REGISTER 1 WILL POINT
                        ;*      TO PHYSICAL 4K - 8K SO "ACT-11" WILL WORK PROPERLY AND MAP
                        ;*      REGISTER 2 WILL POINT TO PHYSICAL 0 - 4K.  THIS MEANS THAT
                        ;*      KIPAR0 SHOULD GET 170400 SO IT PUTS ADDRESSES 040000 TO 057776
                        ;*      ON THE UNIBUS AND KIPAR1 SHOULD GET 170200 SO IT PUTS ADDRESSES
                        ;*      020000 TO 037776 ON THE UNIBUS.
```

```
4178                                    ;*
4179  017760  013701  001246                      MOV     LREGU,R1        ;PUT POINTER TO LOWEST MAP REG IN R1
4180  017764  005061  000004                      CLR     4(R1)           ;CLEAR UPPER 6 BITS OF (LOWEST + 1)
4181  017770  005061  000010                      CLR     10(R1)          ;CLEAR UPPER 6 BITS OF (LOWEST + 2)
4182  017774  012761  020000  000002              MOV     #20000,2(R1)    ;LOAD LOWER 16 BITS OF (LOWEST + 1)
4183                                                                      ;SO THAT IT POINTS TO 4K - 8K
4184  020002  005061  000006                      CLR     6(R1)           ;CLEAR LOWER 16 BITS OF (LOWEST + 2)
4185  020006  013701  001240                      MOV     LOWEST,R1       ;PREPARE TO LOAD PAR1
4186  020012  062701  000200                      ADD     #200,R1         ;POINTER TO (LOWEST + 1)
4187  020016  010137  172342                      MOV     R1,KIPAR1       ;LOAD PAR1 TO POINT TO LOWEST + 1
4188  020022  062701  000200                      ADD     #200,R1         ;ADJUST R1 FOR KIPARO
4189  020026  010137  172340                      MOV     R1,KIPARO       ;LOAD PARO TO POINT TO (LOWEST + 2)
4190
4191
4192                                    ;;*****************************************************************
4193                                    ;*TEST 36       MAIN MEMORY TIMEOUT THRU MAP, CODE RUN OVER UNIBUS
4194                                    ;*
4195                                    ;*      THIS TEST GENERATES A TIME OUT THROUGH THE UNIBUS MAP BY TRYING
4196                                    ;*      TO REFERENCE ADDRESS 17000000 IN MAIN MEMORY.  IT USES THE LOWEST
4197                                    ;*      USEABLE MAP REGISTER, WHICH IN THE DEFAULT CASE IS MAP REGISTER
4198                                    ;*      ZERO.
4199                                    ;*
4200                                    ;*      THIS TEST IS BEING RUN WITH ALL MEMORY REFERENCES GOING THRU
4201                                    ;*      THE UNIBUS MAP.
4202                                    ;*
4203                                    ;;*****************************************************************
4204  020032                            TST36:
4205  020032  012737  020070  001106              MOV     #20$,$LPADR     ;SET LOOP ON TEST POINTER TO 20$
4206  020040  012737  020070  001110              MOV     #20$,$LPERR     ;SET LOOP ON ERROR POINTER TO 20$
4207  020046  012737  000036  001102              MOV     #36,$TSTNM      ;SETUP TEST NUMBER AND CLR ERROR FLAG
4208  020054  013737  001102  177570              MOV     $TSTNM,DISPLAY  ;DISPLAY TEST NUMBER FOR ALL TO SEE
4209  020062  012737  020164  001324              MOV     #TST37,NXTTST   ;SET UP ESCAPE VECTOR IN CASE OF PARITY ERRORS
4210  020070  012737  000020  001264  20$:        MOV     #20,CPUEXP      ;EXPECTING CPU TIMEOUT IN THIS TEST
4211  020076  013737  001240  172350              MOV     LOWEST,KIPAR4   ;LOAD PAR 4 WITH LOWEST USABLE MAP REG
4212  020104  012777  000074  161134              MOV     #74,@LREGU      ;LOAD UPPER 6 BITS OF LOWEST MAP REG
4213  020112  012777  000000  161124              MOV     #000000,@LREGL  ;LOAD LOWER 16 BITS OF LOWEST MAP REG
4214  020120  012737  020126  001110              MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
4215  020126  005037  001266          1$:         CLR     PCPUER          ;CPU ERROR REG LOCATION
4216  020132  000240                              NOP                     ;THIS IS A SYNC POINT FOR SCOPING
4217  020134  013703  100000                      MOV     @#100000,R3     ;TRY TO READ THRU PAGE 4
4218                                                                      ;THIS REFERENCE WILL GO OUT ON THE
4219                                                                      ;UNIBUS TO SELECT THE LOWEST USEABLE
4220                                                                      ;MAP REGISTER (DEFAULT MAP REG. 0).
4221                                                                      ;PHYSICAL ADDRESS 17700000 IS THEN
4222                                                                      ;GENERATED, WHICH SHOULD TIME OUT SINCE
4223                                                                      ;IT IS THE FIRST NON-EXISTANT LOCATION.
4224  020140  022737  000020  001266              CMP     #20,PCPUER      ;THE UNIBUS SHOULD HAVE TIMED OUT
4225  020146  001401                              BEQ     10$             ;BRANCH IF UNIBUS TIMED OUT
4226  020150  104045                              ERROR   45              ;DID NOT TIME OUT OVER UNIBUS
4227  020152  012737  020070  001110  10$:        MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
4228  020160  005037  001264                      CLR     CPUEXP          ;NO CPU TRAPS EXPECTED IN NEXT TEST
4229
4230
4231                                    ;;*****************************************************************
4232                                    ;*TEST 37       RELOCATION TEST USING LOWEST USABLE MAPPING REG
4233                                    ;*
```

```
                                        ;*      THIS TEST CHECKS OUT THE FULL ADDITION PROPERTIES OF THE UNIBUS
                                        ;*      MAP A.L.U..  IN THE DEFAULT CASE IT USES MAP REGISTER ZERO BUT
                                        ;*      IF THE MAP JUMPERS HAVE BEEN ALTERED TO DE-SELECT SOME MAP REGISTERS
                                        ;*      THIS TEST WILL USE THE LOWEST USEABLE MAP REGISTER.
                                        ;*      IF AN ERROR OCCURS THE TEST WILL REPORT THE PHYSICAL ADDRESS
                                        ;*      THAT WAS DESIRED, AND THE DATA AT THE ADDRESS THAT WAS REFERENCED.
                                        ;*
                                        ;*      THIS TEST IS BEING RUN WITH ALL MEMORY REFERENCES GOING THRU
                                        ;*      THE UNIBUS MAP.
                                        ;*
                                        ;**************************************************************
020164                                  TST37:
020164  000004                                SCOPE
020166  012737  021100  001324                MOV     #TST40,NXTTST   ;SAVE STARTING ADDRESS OF NEXT TEST
                                                                        ;FOR ESCAPE ON PARITY ERRORS
020174  012737  060000  060000                MOV     #060000,@#060000  ;MAKE SURE THAT ADDRESS 060000
                                                                        ;CONTAINS ITS OWN ADDRESS AS DATA
                                        ;;
                                        ;*THE RELOCATION HERE USES A BASE OF 00060000 AND AN
                                        ;*OFFSET OF 00000 TO PRODUCE AN ADDRESS OF 00060000
                                        ;;
020202  012737  020234  001110  100$:    MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
020210  005077  161032                    CLR     @LREGU          ;CLEAR UPPER BITS OF MAPPING REG
020214  012777  060000  161022            MOV     #060000,@LREGL  ;LOAD LOWER BITS OF MAPPING REG
020222  013737  001240  172354            MOV     LOWEST,@#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
020230  012700  140000                    MOV     #140000,R0      ;SELECT PAR6, OFFSET IS 00000
020234  000240                  1$:       NOP                     ;THIS IS A SYNC POINT FOR SCOPING
020236  011001                            MOV     (R0),R1         ;READ LOCATION 060000 THRU THE UNIBUS
020240  012702  060000                    MOV     #060000,R2      ;THE EXPECTED PHYSICAL ADDRESS IS 060000
020244  020102                            CMP     R1,R2           ;SEE IF THE MAP'S FETCH WAS CORRECT
020246  001401                            BEQ     2$              ;BRANCH IF FETCHED DATA MATCHES ADDRESS
020250  104046                            ERROR   46              ;FAULTY RELOCATION BY UNIBUS MAP
                                        ;;
                                        ;*THE RELOCATION HERE USES A BASE OF 00052524 AND AN
                                        ;*OFFSET OF 05252 TO PRODUCE AN ADDRESS OF 00057776
                                        ;;
020252  012737  020304  001110  2$:     MOV     #3$,$LPERR      ;SET LOOP ON ERROR POINTER TO 3$
020260  005077  160762                    CLR     @LREGU          ;CLEAR UPPER BITS OF MAPPING REG
020264  012777  052524  160752            MOV     #052524,@LREGL  ;LOAD LOWER BITS OF MAPPING REG
020272  013737  001240  172354            MOV     LOWEST,@#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
020300  012700  145252                    MOV     #145252,R0      ;SELECT PAR6, OFFSET IS 05252
020304  000240                  3$:       NOP                     ;THIS IS A SYNC POINT FOR SCOPING
020306  011001                            MOV     (R0),R1         ;READ LOCATION 057776 THRU THE UNIBUS
020310  012702  057776                    MOV     #057776,R2      ;THE EXPECTED PHYSICAL ADDRESS IS 057776
020314  020102                            CMP     R1,R2           ;SEE IF THE MAP'S FETCH WAS CORRECT
020316  001401                            BEQ     4$              ;BRANCH IF FETCHED DATA MATCHES ADDRESS
020320  104046                            ERROR   46              ;FAULTY RELOCATION BY UNIBUS MAP
                                        ;;
                                        ;*THE RELOCATION HERE USES A BASE OF 00045252 AND AN
                                        ;*OFFSET OF 12524 TO PRODUCE AN ADDRESS OF 00057776
                                        ;;
020322  012737  020354  001110  4$:     MOV     #5$,$LPERR      ;SET LOOP ON ERROR POINTER TO 5$
020330  005077  160712                    CLR     @LREGU          ;CLEAR UPPER BITS OF MAPPING REG
020334  012777  045252  160702            MOV     #045252,@LREGL  ;LOAD LOWER BITS OF MAPPING REG
020342  013737  001240  172354            MOV     LOWEST,@#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
020350  012700  152524                    MOV     #152524,R0      ;SELECT PAR6, OFFSET IS 12524
```

```
4290  020354  000240                  5$:     NOP                     ;THIS IS A SYNC POINT FOR SCOPING
4291  020356  011001                          MOV     (R0),R1         ;READ LOCATION 057776 THRU THE UNIBUS
4292  020360  012702  057776                  MOV     #057776,R2      ;THE EXPECTED PHYSICAL ADDRESS IS 057776
4293  020364  020102                          CMP     R1,R2           ;SEE IF THE MAP'S FETCH WAS CORRECT
4294  020366  001401                          BEQ     6$              ;BRANCH IF FETCHED DATA MATCHES ADDRESS
4295  020370  104046                          ERROR   46              ;FAULTY RELOCATION BY UNIBUS MAP
4296                                  ;;
4297                                  ;*THE RELOCATION HERE USES A BASE OF 00050420 AND AN
4298                                  ;*OFFSET OF 10420 TO PRODUCE AN ADDRESS OF 00061040
4299                                  ;;
4300  020372  012737  020424  001110  6$:     MOV     #7$,$LPERR      ;SET LOOP ON ERROR POINTER TO 7$
4301  020400  005077  160642                  CLR     @LREGU          ;CLEAR UPPER BITS OF MAPPING REG
4302  020404  012777  050420  160632          MOV     #050420,@LREGL  ;LOAD LOWER BITS OF MAPPING REG
4303  020412  013737  001240  172354          MOV     LOWEST,@#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
4304  020420  012700  150420                  MOV     #150420,R0      ;SELECT PAR6, OFFSET IS 10420
4305  020424  000240                  7$:     NOP                     ;THIS IS A SYNC POINT FOR SCOPING
4306  020426  011001                          MOV     (R0),R1         ;READ LOCATION 061040 THRU THE UNIBUS
4307  020430  012702  061040                  MOV     #061040,R2      ;THE EXPECTED PHYSICAL ADDRESS IS 061040
4308  020434  020102                          CMP     R1,R2           ;SEE IF THE MAP'S FETCH WAS CORRECT
4309  020436  001401                          BEQ     8$              ;BRANCH IF FETCHED DATA MATCHES ADDRESS
4310  020440  104046                          ERROR   46              ;FAULTY RELOCATION BY UNIBUS MAP
4311                                  ;;
4312                                  ;*THE RELOCATION HERE USES A BASE OF 00054630 AND AN
4313                                  ;*OFFSET OF 04210 TO PRODUCE AN ADDRESS OF 00061040
4314                                  ;;
4315  020442  012737  020474  001110  8$:     MOV     #9$,$LPERR      ;SET LOOP ON ERROR POINTER TO 9$
4316  020450  005077  160572                  CLR     @LREGU          ;CLEAR UPPER BITS OF MAPPING REG
4317  020454  012777  054630  160562          MOV     #054630,@LREGL  ;LOAD LOWER BITS OF MAPPING REG
4318  020462  013737  001240  172354          MOV     LOWEST,@#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
4319  020470  012700  144210                  MOV     #144210,R0      ;SELECT PAR6, OFFSET IS 04210
4320  020474  000240                  9$:     NOP                     ;THIS IS A SYNC POINT FOR SCOPING
4321  020476  011001                          MOV     (R0),R1         ;READ LOCATION 061040 THRU THE UNIBUS
4322  020500  012702  061040                  MOV     #061040,R2      ;THE EXPECTED PHYSICAL ADDRESS IS 061040
4323  020504  020102                          CMP     R1,R2           ;SEE IF THE MAP'S FETCH WAS CORRECT
4324  020506  001401                          BEQ     10$             ;BRANCH IF FETCHED DATA MATCHES ADDRESS
4325  020510  104046                          ERROR   46              ;FAULTY RELOCATION BY UNIBUS MAP
4326                                  ;;
4327                                  ;*THE RELOCATION HERE USES A BASE OF 00044210 AND AN
4328                                  ;*OFFSET OF 14630 TO PRODUCE AN ADDRESS OF 00061040
4329                                  ;;
4330  020512  012737  020544  001110  10$:    MOV     #11$,$LPERR     ;SET LOOP ON ERROR POINTER TO 11$
4331  020520  005077  160522                  CLR     @LREGU          ;CLEAR UPPER BITS OF MAPPING REG
4332  020524  012777  044210  160512          MOV     #044210,@LREGL  ;LOAD LOWER BITS OF MAPPING REG
4333  020532  013737  001240  172354          MOV     LOWEST,@#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
4334  020540  012700  154630                  MOV     #154630,R0      ;SELECT PAR6, OFFSET IS 14630
4335  020544  000240                  11$:    NOP                     ;THIS IS A SYNC POINT FOR SCOPING
4336  020546  011001                          MOV     (R0),R1         ;READ LOCATION 061040 THRU THE UNIBUS
4337  020550  012702  061040                  MOV     #061040,R2      ;THE EXPECTED PHYSICAL ADDRESS IS 061040
4338  020554  020102                          CMP     R1,R2           ;SEE IF THE MAP'S FETCH WAS CORRECT
4339  020556  001401                          BEQ     12$             ;BRANCH IF FETCHED DATA MATCHES ADDRESS
4340  020560  104046                          ERROR   46              ;FAULTY RELOCATION BY UNIBUS MAP
4341                                  ;;
4342                                  ;*THE RELOCATION HERE USES A BASE OF 00056734 AND AN
4343                                  ;*OFFSET OF 02104 TO PRODUCE AN ADDRESS OF 00061040
4344                                  ;;
4345  020562  012737  020614  001110  12$:    MOV     #13$,$LPERR     ;SET LOOP ON ERROR POINTER TO 13$
```

```
4346  020570  005077  160452                  CLR     @LREGU          ;CLEAR UPPER BITS OF MAPPING REG
4347  020574  012777  056734  160442          MOV     #056734,@LREGL  ;LOAD LOWER BITS OF MAPPING REG
4348  020602  013737  001240  172354          MOV     LOWEST,@#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
4349  020610  012700  142104                  MOV     #142104,R0      ;SELECT PAR6, OFFSET IS 02104
4350  020614  000240                  13$:    NOP                     ;THIS IS A SYNC POINT FOR SCOPING
4351  020616  011001                          MOV     (R0),R1         ;READ LOCATION 061040 THRU THE UNIBUS
4352  020620  012702  061040                  MOV     #061040,R2      ;THE EXPECTED PHYSICAL ADDRESS IS 061040
4353  020624  020102                          CMP     R1,R2           ;SEE IF THE MAP'S FETCH WAS CORRECT
4354  020626  001401                          BEQ     14$             ;BRANCH IF FETCHED DATA MATCHES ADDRESS
4355  020630  104046                          ERROR   46              ;FAULTY RELOCATION BY UNIBUS MAP
4356                                  ;;
4357                                  ;*THE RELOCATION HERE USES A BASE OF 00042104 AND AN
4358                                  ;*OFFSET OF 16734 TO PRODUCE AN ADDRESS OF 00061040
4359                                  ;;
4360  020632  012737  020664  001110  14$:    MOV     #15$,$LPERR     ;SET LOOP ON ERROR POINTER TO 15$
4361  020640  005077  160402                  CLR     @LREGU          ;CLEAR UPPER BITS OF MAPPING REG
4362  020644  012777  042104  160372          MOV     #042104,@LREGL  ;LOAD LOWER BITS OF MAPPING REG
4363  020652  013737  001240  172354          MOV     LOWEST,@#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
4364  020660  012700  156734                  MOV     #156734,R0      ;SELECT PAR6, OFFSET IS 16734
4365  020664  000240                  15$:    NOP                     ;THIS IS A SYNC POINT FOR SCOPING
4366  020666  011001                          MOV     (R0),R1         ;READ LOCATION 061040 THRU THE UNIBUS
4367  020670  012702  061040                  MOV     #061040,R2      ;THE EXPECTED PHYSICAL ADDRESS IS 061040
4368  020674  020102                          CMP     R1,R2           ;SEE IF THE MAP'S FETCH WAS CORRECT
4369  020676  001401                          BEQ     16$             ;BRANCH IF FETCHED DATA MATCHES ADDRESS
4370  020700  104046                          ERROR   46              ;FAULTY RELOCATION BY UNIBUS MAP
4371                                  ;;
4372                                  ;*THE RELOCATION HERE USES A BASE OF 00057776 AND AN
4373                                  ;*OFFSET OF 01042 TO PRODUCE AN ADDRESS OF 00061040
4374                                  ;;
4375  020702  012737  020734  001110  16$:    MOV     #17$,$LPERR     ;SET LOOP ON ERROR POINTER TO 17$
4376  020710  005077  160332                  CLR     @LREGU          ;CLEAR UPPER BITS OF MAPPING REG
4377  020714  012777  057776  160322          MOV     #057776,@LREGL  ;LOAD LOWER BITS OF MAPPING REG
4378  020722  013737  001240  172354          MOV     LOWEST,@#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
4379  020730  012700  141042                  MOV     #141042,R0      ;SELECT PAR6, OFFSET IS 01042
4380  020734  000240                  17$:    NOP                     ;THIS IS A SYNC POINT FOR SCOPING
4381  020736  011001                          MOV     (R0),R1         ;READ LOCATION 061040 THRU THE UNIBUS
4382  020740  012702  061040                  MOV     #061040,R2      ;THE EXPECTED PHYSICAL ADDRESS IS 061040
4383  020744  020102                          CMP     R1,R2           ;SEE IF THE MAP'S FETCH WAS CORRECT
4384  020746  001401                          BEQ     18$             ;BRANCH IF FETCHED DATA MATCHES ADDRESS
4385  020750  104046                          ERROR   46              ;FAULTY RELOCATION BY UNIBUS MAP
4386                                  ;;
4387                                  ;*THE RELOCATION HERE USES A BASE OF 00041042 AND AN
4388                                  ;*OFFSET OF 17776 TO PRODUCE AN ADDRESS OF 00061040
4389                                  ;;
4390  020752  012737  021004  001110  18$:    MOV     #19$,$LPERR     ;SET LOOP ON ERROR POINTER TO 19$
4391  020760  005077  160262                  CLR     @LREGU          ;CLEAR UPPER BITS OF MAPPING REG
4392  020764  012777  041042  160252          MOV     #041042,@LREGL  ;LOAD LOWER BITS OF MAPPING REG
4393  020772  013737  001240  172354          MOV     LOWEST,@#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
4394  021000  012700  157776                  MOV     #157776,R0      ;SELECT PAR6, OFFSET IS 17776
4395  021004  000240                  19$:    NOP                     ;THIS IS A SYNC POINT FOR SCOPING
4396  021006  011001                          MOV     (R0),R1         ;READ LOCATION 061040 THRU THE UNIBUS
4397  021010  012702  061040                  MOV     #061040,R2      ;THE EXPECTED PHYSICAL ADDRESS IS 061040
4398  021014  020102                          CMP     R1,R2           ;SEE IF THE MAP'S FETCH WAS CORRECT
4399  021016  001401                          BEQ     20$             ;BRANCH IF FETCHED DATA MATCHES ADDRESS
4400  021020  104046                          ERROR   46              ;FAULTY RELOCATION BY UNIBUS MAP
4401                                  ;;
```

```
                                        ;*THE RELOCATION HERE USES A BASE OF 00057776 AND AN
                                        ;*OFFSET OF 00002 TO PRODUCE AN ADDRESS OF 00060000
                                        ;;
021022  012737  021054  001110  20$:    MOV     #21$,$LPERR     ;SET LOOP ON ERROR POINTER TO 21$
021030  005077  160212                  CLR     @LREGU          ;CLEAR UPPER BITS OF MAPPING REG
021034  012777  057776  160202          MOV     #057776,@LREGL  ;LOAD LOWER BITS OF MAPPING REG
021042  013737  001240  172354          MOV     LOWEST,@#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
021050  012700  140002                  MOV     #140002,R0      ;SELECT PAR6, OFFSET IS 00002
021054  000240                  21$:    NOP                     ;THIS IS A SYNC POINT FOR SCOPING
021056  011001                          MOV     (R0),R1         ;READ LOCATION 060000 THRU THE UNIBUS
021060  012702  060000                  MOV     #060000,R2      ;THE EXPECTED PHYSICAL ADDRESS IS 060000
021064  020102                          CMP     R1,R2           ;SEE IF THE MAP'S FETCH WAS CORRECT
021066  001401                          BEQ     22$             ;BRANCH IF FETCHED DATA MATCHES ADDRESS
021070  104046                          ERROR   46              ;FAULTY RELOCATION BY UNIBUS MAP
021072  012737  020202  001110  22$:    MOV     #100$,$LPERR    ;SET LOOP POINTER TO START OF TEST
                                        ;;*****************************************************************
                                        ;*TEST 40       TEST CARRY PROPAGATION OF MAP'S RELOCATION ADDER
                                        ;*
                                        ;*      EVERY ADDRESS OF THE FORM XXXX0000 IS GENERATED HERE STARTING
                                        ;*      WITH 00030000 UP TO 17000000.  THAT IS THE FIRST OF EVERY 2K
                                        ;*      WORDS IS ADDRESSED, TO INSURE THAT THE ADDER IN THE MAP IS
                                        ;*      WORKING PROPERLY AND, THE SYSTEM SIZE JUMPERS ARE ACTUALLY SET
                                        ;*      FOR THE TOP OF MAIN MEMORY.
                                        ;*
                                        ;;*****************************************************************
021100                          TST40:
021100  000004                          SCOPE
021102  012737  021370  001324          MOV     #TST41,NXTTST   ;SAVE STARTING ADDRESS OF NEXT TEST
                                                                ;FOR ESCAPE ON PARITY ERRORS
021110  012737  000012  001204          MOV     #12,$TIMES      ;;DO 12 ITERATIONS
021116  012737  177777  021252  20$:    MOV     #-1,4$          ;INITIALIZE FLAG AS NEGATIVE ONE
021124  005077  160116                  CLR     @LREGU          ;CLEAR UPPER 6 BITS OF MAP REG
021130  012777  020000  160106          MOV     #20000,@LREGL   ;LOAD 4K BASE INTO MAP REGISTER
021136  012701  100100                  MOV     #100100,R1      ;LOAD BITS TO SELECT PAR 4, OFFSET 100
021142  012700  150000                  MOV     #150000,R0      ;LOAD BITS TO SELECT PAR 6, OFFSET 2K
021146  012737  000277  172350          MOV     #277,KIPAR4     ;START WITH PHYSICAL 6K
021154  013737  001240  172354          MOV     LOWEST,KIPAR6   ;LOAD PAR 6 WITH MAP REG'S ADDR
021162  012737  021222  001110          MOV     #10$,$LPERR     ;SET LOOP ON ERROR POINTER TO 10$
021170  012737  000020  001264  3$:     MOV     #TIMOUT,CPUEXP  ;EXPECTING A UNIBUS TIME OUT DURING TEST
021176  005037  001266                  CLR     PCPUER          ;CLEAR TIME OUT FLAG
021202  013710  001326                  MOV     DATA,(R0)       ;THIS LOAD WILL TIME OUT WHEN YOU
                                                                ;HAVE REACHED THE TOP OF MEMORY
                                                                ;IT SELECTS PAR 6 WHICH WILL PUT ADDR
                                                                ;<XXX XX1>0000 ON THE UNIBUS.
                                                                ;THE X'S WILL SELECT THE LOWEST USEABLE
                                                                ;MAPPING REGISTER.  THE DEFAULT CASE IS
                                                                ;010000,  SELECTING MAPPING REGISTER 0.
021206  005737  001266                  TST     PCPUER          ;SEE IF THERE WAS MAIN MEMORY
021212  001016                          BNE     1$              ;BRANCH IF NO MAIN MEMORY FROM UNIBUS
021214  012737  000040  001264          MOV     #NEXMEM,CPUEXP  ;POSSIBLE CACHE NON-EXISTANT MEMORY
021222  011103                  10$:    MOV     (R1),R3         ;READ TEST LOCATION VIA FASTBUS
021224  022737  000040  001266          CMP     #NEXMEM,PCPUER  ;WAS THERE CACHE NON-EXISTANT MEMORY
```

```
4458  021232  001414                          BEQ     2$              ;BRANCH IF NON-EXISTANT MEMORY
4459  021234  000240                          NOP                     ;THIS IS A SYNC POINT FOR SCOPING
4460  021236  011002                          MOV     (R0),R2         ;READ TEST LOCATION VIA UNIBUS MAP
4461  021240  020203                          CMP     R2,R3           ;COMPARE TEST DATA R2=MAP DATA
4462                                                                  ;R3=FASTBUS DATA
4463  021242  001410                          BEQ     2$              ;BRANCH IF IT WAS THE SAME
4464  021244  104047                          ERROR   47              ;FAILURE IN CARRY PROPAGATION IN MAP
4465  021246  000406                          BR      2$              ;BRANCH TO UPDATE ROUTINE
4466  021250  005227                  1$:     INC     (PC)+           ;INCREMENT ONE TIME ENTRANCE FLAG
4467  021252  177777                  4$:     .WORD   -1              ;USE NEGATIVE ONE FOR FLAG
4468  021254  001003                          BNE     2$              ;BRANCH IF YOU'VE BEEN HERE BEFORE
4469  021256  013737  172350  001320          MOV     KIPAR4,RSIZE    ;SAVE UPPER LIMIT OF MEMORY
4470  021264  062737  000100  001326  2$:     ADD     #100,DATA       ;CHANGE PATTERN FOR NEXT LOAD
4471  021272  062737  000100  172350          ADD     #100,KIPAR4     ;ADD 2K TO PAR4
4472  021300  062777  010000  157736          ADD     #10000,@LREGL   ;ADD 2K TO MAP REGISTER
4473  021306  001330                          BNE     3$              ;BRANCH IF MAP REGISTER NOT ZERO
4474  021310  005277  157732                  INC     @LREGU          ;ADD ONE TO UPPER 6 BITS OF MAP REG
4475  021314  022777  000073  157724          CMP     #73,@LREGU      ;SEE IF TOP 128K BLOCK HAS BEEN PASSED
4476  021322  103322                          BHIS    3$              ;BRANCH IF NOT PAST IT
4477  021324  005237  001326                  INC     DATA            ;CHANGE DATA PATTERN FOR NEXT PASS
4478  021330  042737  177700  001326          BIC     #177700,DATA    ;CLEAR UPPER 10 BITS OF DATA PATTERN
4479  021336  052737  000300  001326          BIS     #300,DATA       ;START WITH 3XX IN DATA PATTERN
4480  021344  012737  021116  001110          MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
4481  021352  023737  177760  001320          CMP     @#SIZELO,RSIZE  ;SEE IF SIZE JUMPERS AGREE WITH MEMORY TOP
4482  021360  001401                          BEQ     19$             ;BRANCH IF TOP OF MEMORY AGREES WITH
4483                                                                  ;THE SIZE JUMPERS
4484  021362  104050                          ERROR   50              ;MEMORY SIZE JUMPERS NOT RIGHT
4485  021364  005037  001264          19$:    CLR     CPUEXP          ;NO CPU TRAPS EXPECTED IN NEXT TEST
4486
4487
4488
4489                                  ;;*****************************************************************
4490                                  ;*TEST 41       PARITY REPORTING THRU THE MAP, MAIN MEMORY EVEN WORD
4491                                  ;*
4492                                  ;*      THIS TEST FORCES A PARITY ERROR IN MAIN MEMORY AT PHYSICAL
4493                                  ;*      ADDRESS 00040000 BY SETTING ONE BIT IN EACH BYTE OF THE
4494                                  ;*      WORD.  THE TEST THEN FORCES THE EVEN WORD PARITY BITS TO
4495                                  ;*      ONES AND READS ADDRESS 00040000 THRU THE UNIBUS MAP.  THEN
4496                                  ;*      IT CHECKS TO SEE THAT THE PARITY ABORT OCCURRED AND THAT THE
4497                                  ;*      CONDITION WAS CORRECT BEFORE GOING TO THE NEXT TEST.
4498                                  ;*
4499                                  ;*      ERRORS ARE REPORTED INLINE IN THE TEST CODE, AND THE PARITY
4500                                  ;*      VECTOR WILL TRAP TO LABEL 10$ AT THE END OF THIS TEST.
4501                                  ;*
4502                                  ;*****************************************************************
4503  021370                          TST41:
4504  021370  000004                          SCOPE
4505  021372  012737  021632  001324          MOV     #TST42,NXTTST   ;SAVE STARTING ADDRESS OF NEXT TEST
4506                                                                  ;FOR ESCAPE ON PARITY ERRORS
4507  021400  104416                          TBITO                   ;TURN OFF T BIT TRAPPING IF ON
4508  021402  012737  023404  001200  20$:     MOV     #23404,$TMP4    ;EXPECTED ERROR CONDITION
4509  021410  005037  001274                  CLR     PPARER          ;CLEAR MEM ERROR REG STORAGE
4510  021414  012737  177777  177744          MOV     #-1,@#MEMERR    ;CLEAR MEMORY ERROR REGISTER
4511  021422  013737  177746  001202          MOV     @#CONTRL,$TMP5  ;SAVE CONTROL REG TO RESTORE CACHE
4512  021430  012737  000014  177746          MOV     #14,@#CONTRL    ;FORCE MISSES IN CACHE
4513  021436  013737  001240  172354          MOV     LOWEST,KIPAR6   ;PUT UNIBUS ADDR OF MAP REG IN PAR6
```

```
4514  021444  012777  040000  157572          MOV     #40000,@LREGL   ;LOWEST MAP REGISTER POINTS TO 8K
4515  021452  005077  157570                  CLR     @LREGU          ;CLEAR UPPER BITS OF MAP REG
4516  021456  012701  177750                  MOV     #MAINT,R1       ;PUT ADDR OF MAINT REG IN R1
4517  021462  012737  021500  001110          MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
4518  021470  012700  140000                  MOV     #140000,R0      ;SELECT KIPAR6, OFFSET 0
4519  021474  012710  020001                  MOV     #20001,(R0)     ;LOAD ONE BIT IN EACH BYTE OF 40000
4520  021500  012737  021572  000114  1$:     MOV     #10$,CACHVEC    ;SET PARITY VECTOR TO 10$
4521  021506  012737  030000  177750          MOV     #30000,@#MAINT  ;FORCE EVEN WORD PARITY BITS TO ONE
4522  021514  000240                          NOP                     ;THIS IS A SYNC POINT FOR SCOPING
4523  021516  011003                          MOV     (R0),R3         ;"DATA FETCH" SHOULD CAUSE PARITY ABORT
4524  021520  011102                          MOV     (R1),R2         ;SAVE MAINT REG IN CASE NO TRAP
4525  021522  005011                          CLR     (R1)            ;CLEAR MAINT REG IN CASE NO TRAP
4526  021524  050000                          BIS     R0,R0           ;DUMMY INST WITH P.B.'S ON
4527  021526  023737  001200  001274          CMP     $TMP4,PPARER    ;SEE IF ERROR REG HOLDS RIGHT DATA
4528  021534  001401                          BEQ     2$              ;BRANCH IF CONDITION WAS CORRECT
4529  021536  104051                          ERROR   51              ;FAILURE IN PARITY REPORTING
4530  021540  012737  177777  177744  2$:     MOV     #-1,MEMERR      ;CLEAR MEMORY ERROR REGISTER
4531  021546  012737  005704  000114          MOV     #MEMER,CACHVEC  ;RESTORE PARITY SERVICE ROUTINE
4532  021554  013737  001202  177746          MOV     $TMP5,CONTRL    ;RESTORE CACHE TO FORMER CONDITION
4533  021562  012737  021402  001110          MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
4534  021570  000420                          BR      TST42           ;;TEST OVER GO TO NEXT TEST
4535
4536
4537  021572  013737  177740  001270  10$:     MOV     @#LOADRS,PLOADR ;SAVE LOWER CACHE ADDR REG
4538  021600  013737  177742  001272          MOV     @#HIADRS,PHIADR ;SAVE HIGH BITS OF FAILING ADDR
4539  021606  013737  177744  001274          MOV     @#MEMERR,PPARER ;SAVE MEMORY ERROR REGISTER
4540  021614  013737  177746  001276          MOV     @#CONTRL,PCONTR ;SAVE CONTROL REGISTER FOR TYPE OUT
4541  021622  012737  030000  001300          MOV     #30000,PMAINT   ;SAVE DATA IN MAINTENENCE REGISTER
4542  021630  000002                          RTI                     ;RETURN TO TEST AND CHECK PARITY
4543
4544
4545                                  ;;*****************************************************************
4546                                  ;*TEST 42       PARITY REPORTING THRU THE MAP, MAIN MEMORY ODD WORD
4547                                  ;*
4548                                  ;*      THIS TEST FORCES A PARITY ERROR IN MAIN MEMORY AT PHYSICAL
4549                                  ;*      ADDRESS 00040002 BY SETTING ONE BIT IN EACH BYTE OF THE
4550                                  ;*      WORD.  THE TEST THEN FORCES THE ODD WORD PARITY BITS TO
4551                                  ;*      ONES AND READS ADDRESS 00040002 THRU THE UNIBUS MAP.  THEN
4552                                  ;*      IT CHECKS TO SEE THAT THE PARITY ABORT OCCURRED AND THAT THE
4553                                  ;*      CONDITION WAS CORRECT BEFORE GOING TO THE NEXT TEST.
4554                                  ;*
4555                                  ;*      ERRORS ARE REPORTED INLINE IN THE TEST CODE, AND THE PARITY
4556                                  ;*      VECTOR WILL TRAP TO LABEL 10$ AT THE END OF THIS TEST.
4557                                  ;*
4558                                  ;*      THIS TEST IS BEING RUN WITH ALL MEMORY REFERENCES GOING THRU
4559                                  ;*      THE UNIBUS MAP.
4560                                  ;*
4561                                  ;;*****************************************************************
4562  021632                          TST42:
4563  021632  000004                          SCOPE
4564  021634  012737  022104  001324          MOV     #TST43,NXTTST   ;SAVE STARTING ADDRESS OF NEXT TEST
4565                                                                  ;FOR ESCAPE ON PARITY ERRORS
4566  021642  012737  023410  001200  20$:     MOV     #23410,$TMP4    ;LOAD EXPECTED ERROR CONDITION
4567  021650  005037  001274                  CLR     PPARER          ;CLEAR MEM ERROR REG STORAGE
4568  021654  012737  177777  177744          MOV     #-1,@#MEMERR    ;CLEAR MEMORY ERROR REGISTER
4569  021662  013737  177746  001202          MOV     @#CONTRL,$TMP5  ;SAVE CONTROL REG TO RESTORE CACHE
```

```
 4570  021670  012737  000014  177746          MOV     #14,@#CONTRL    ;FORCE MISSES IN CACHE
 4571  021676  013737  001240  172354          MOV     LOWEST,KIPAR6   ;PUT UNIBUS ADDR OF MAP REG IN PAR6
 4572  021704  012777  040000  157332          MOV     #40000,@LREGL   ;LOWEST MAP REGISTER POINTS TO 8K
 4573  021712  005077  157330                  CLR     @LREGU          ;CLEAR UPPER BITS OF MAP REG
 4574  021716  012701  177750                  MOV     #MAINT,R1       ;PUT ADDR OF MAINT REG IN R1
 4575  021722  012737  021752  001110          MOV     #1$,$LPERR      ;SET LOOP ON EROR POINTER TO 1$
 4576  021730  005037  140000                  CLR     140000          ;CLEAR LOCATION 40000
 4577  021734  012700  140002                  MOV     #140002,R0      ;LOAD ADDRESS OF 40002 INTO R0
 4578  021740  012737  021752  001110          MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
 4579  021746  012710  020001                  MOV     #20001,(R0)     ;LOAD ONE BIT IN EACH BYTE OF 40002
 4580  021752  012737  022044  000114  1$:     MOV     #10$,CACHVEC    ;SET PARITY VECTOR TO 10$
 4581  021760  012737  140000  177750          MOV     #140000,@#MAINT ;FORCE ODD WORD PARITY BITS TO ONE
 4582  021766  000240                          NOP                     ;THIS IS A SYNC POINT FOR SCOPING
 4583  021770  011003                          MOV     (R0),R3         ;"DATA FETCH" SHOULD CAUSE PARITY ABORT
 4584  021772  011102                          MOV     (R1),R2         ;SAVE MAINT REG IN CASE NO TRAP
 4585  021774  005011                          CLR     (R1)            ;CLEAR MAINT REG IN CASE NO TRAP
 4586  021776  050000                          BIS     R0,R0           ;DUMMY INST WITH P.B.'S ON
 4587  022000  023737  001200  001274          CMP     $TMP4,PPARER    ;SEE IF MEM ERROR REG HOLDS RIGHT DATA
 4588  022006  001401                          BEQ     2$              ;BRANCH IF ERROR WAS EXPECTED ONE
 4589  022010  104051                          ERROR   51              ;FAILURE IN PARITY REPORTING
 4590  022012  012737  177777  177744  2$:     MOV     #-1,MEMERR      ;CLEAR MEMORY ERROR REGISTER
 4591  022020  012737  005704  000114          MOV     #MEMER,CACHVEC  ;RESTORE PARITY SERVICE ROUTINE
 4592  022026  013737  001202  177746          MOV     $TMP5,CONTRL    ;RESTORE CACHE TO FORMER CONDITION
 4593  022034  012737  021642  001110          MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
 4594  022042  000420                          BR      TST43           ;;TEST IS OVER, GO TO NEXT TEST
 4595
 4596
 4597  022044  013737  177740  001270  10$:     MOV     @#LOADRS,PLOADR ;SAVE LOWER CACHE ADDR REG
 4598  022052  013737  177742  001272          MOV     @#HIADRS,PHIADR ;SAVE HIGH BITS OF FAILING ADDR
 4599  022060  013737  177744  001274          MOV     @#MEMERR,PPARER ;SAVE MEMORY ERROR REGISTER
 4600  022066  013737  177746  001276          MOV     @#CONTRL,PCONTR ;SAVE CONTROL REGISTER FOR TYPE OUT
 4601  022074  012737  140000  001300          MOV     #140000,PMAINT  ;SAVE DATA IN MAINTENENCE REGISTER
 4602  022102  000002                          RTI                     ;RETURN TO TEST AND CHECK PARITY
 4603
 4604
 4605                                  ;;*****************************************************************
 4606                                  ;*TEST 43       PARITY REPORTING THRU THE MAP, CACHE DATA GROUP 0
 4607                                  ;*
 4608                                  ;*      THIS TEST FORCES A PARITY ERROR IN CACHE GROUP 0 AT PHYSICAL
 4609                                  ;*      ADDRESS 00040000 BY SETTING BIT 15 IN THE WORD.  THE TEST
 4610                                  ;*      THEN FORCES GROUP 0 LOW BYTE PARITY BIT TO ZERO AND EVEN
 4611                                  ;*      WORD HIGH BYTE PARITY BIT TO ONE BEFORE READING ADDRESS 00040000
 4612                                  ;*      THRU THE UNIBUS MAP.  THEN IT CHECKS TO SEE THAT THE PARITY
 4613                                  ;*      ABORT OCCURRED AND THAT THE CONDITION WAS CORRECT BEFORE
 4614                                  ;*      GOING TO THE NEXT TEST.
 4615                                  ;*
 4616                                  ;*      ERRORS ARE REPORTED INLINE IN THE TEST CODE, AND THE PARITY
 4617                                  ;*      VECTOR WILL TRAP TO LABEL 10$ AT THE END OF THIS TEST.
 4618                                  ;*
 4619                                  ;*      THIS TEST IS BEING RUN WITH ALL MEMORY REFERENCES GOING THRU
 4620                                  ;*      THE UNIBUS MAP.
 4621                                  ;*
 4622                                  ;;*****************************************************************
 4623  022104                          TST43:
 4624  022104  000004                          SCOPE
 4625  022106  012737  022372  001324          MOV     #TST44,NXTTST   ;SAVE STARTING ADDRESS OF NEXT TEST
```

```
                                                        ;FOR ESCAPE ON PARITY ERRORS
022114  012737  177777  022260  20$:    MOV     #-1,21$ ;RESTORE NEG ONE FOR NEXT ITERATION
022122  032737  000004  177746          BIT     #BIT2,@#CONTRL  ;SEE IF GROUP 0 IS DISABLED
022130  001120                        BNE     TST44   ;;BRANCH IF GROUP 0 IS DISABLED
022132  013737  177746  001202        MOV     CONTRL,$TMP5    ;SAVE CONDITION OF CACHE
022140  012737  023504  001200        MOV     #23504,$TMP4    ;EXPECTED PARITY CONDITION GROUP 0
022146  005037  001274                CLR     PPARER  ;CLEAR MEM ERROR REG STORAGE
022152  012737  177777  177744        MOV     #-1,@#MEMERR    ;CLEAR MEMORY ERROR REGISTER
022160  013737  001240  172354        MOV     LOWEST,KIPAR6   ;PUT UNIBUS ADDR OF MAP REG IN PAR6
022166  005077  157054                CLR     @LREGU  ;CLEAR UPPER BITS ON MAP REG
022172  012777  040000  157044        MOV     #40000,@LREGL   ;LOAD LOWER 16 BITS OF MAP REG
022200  012705  177750                MOV     #MAINT,R5       ;PUT ADDR OF MAINT REG IN R5
022204  012704  140000                MOV     #140000,R4      ;SELECT PAR 6 BASE 0
022210  012714  100000                MOV     #BIT15,(R4)     ;MAKE SURE I GET SOFT AND HARD ERRORS
                                                        ;BY LOADING 40000 WITH BIT 15, THIS
                                                        ;CAUSES THE HIGH BYTE P.B. TO BE ON
                                                        ;AND THE LOW BYTE P.B. TO BE OFF
022214  012737  000030  177746        MOV     #30,@#CONTRL    ;FORCE SELECTION OF GROUP 0 ON READ
022222  011401                        MOV     (R4),R1 ;PUT (40000) & (40002) IN GROUP 0
022224  005001                        CLR     R1      ;ZERO R1 FOR THE MOMENT
022226  012737  022332  000114        MOV     #10$,CACHVEC    ;SET PARITY VECTOR TO 10$
022234  010137  177750          1$:   MOV     R1,@#MAINT      ;FORCE GPO PARITY BITS TO 0 ON 2ND PASS
022240  000241                        CLC                     ;THIS IS A SYNC POINT FOR SCOPING
                                                        ;THE LOW BYTE HAS ITS P.B. OFF
022242  011401                        MOV     (R4),R1 ;DATA FETCH SHOULD CAUSE PARITY ABORT
022244  011503                        MOV     (R5),R3 ;SAVE MAINT REG IN CASE NO TRAP
022246  105015                        CLRB    (R5)    ;CLEAR MAINT REG IN CASE NO TRAP
022250  006701                        SXT     R1      ;DUMMY INST WITH P.B.'S OFF
022252  012701  020020                MOV     #020020,R1      ;FORCE GROUP 0 LOW BYTE PARITY BIT
                                                        ;TO 0 AND MAIN MEMORY EVEN WORD HIGH
                                                        ;BYTE PARITY BIT TO 1
022256  005227                        INC     (PC)+   ;INCREMENT NEXT WORD TO ZERO ON FIRST PASS
022260  177777          21$:  .WORD   -1      ;
022262  001764                        BEQ     1$      ;BRANCH ON FIRST PASS ONLY
022264  023737  001200  001274        CMP     $TMP4,PPARER    ;SEE IF ERROR CONDITION MATCHES EXPECTED
022272  001402                        BEQ     2$      ;BRANCH IF CORRECT CONDITION
022274  010302                        MOV     R3,R2   ;R2 HOLDS MAINT REG FOR ERROR TYPE OUT
022276  104051                        ERROR   51      ;FAILURE IN PARITY REPORTING
022300  012737  177777  177744  2$:   MOV     #-1,MEMERR      ;CLEAR MEMORY ERROR REGISTER
022306  012737  005704  000114        MOV     #MEMER,CACHVEC  ;RESTORE PARITY SERVICE ROUTINE
022314  013737  001202  177746        MOV     $TMP5,CONTRL    ;RESTORE CACHE TO FORMER CONDITION
022322  012737  022114  001110        MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
022330  000420                        BR      TST44   ;;TEST OVER GO TO NEXT TEST


022332  013737  177740  001270  10$:   MOV     @#LOADRS,PLOADR ;SAVE LOWER CACHE ADDR REG
022340  013737  177742  001272        MOV     @#HIADRS,PHIADR ;SAVE HIGH BITS OF FAILING ADDR
022346  013737  177744  001274        MOV     @#MEMERR,PPARER ;SAVE MEMORY ERROR REGISTER
022354  013737  177746  001276        MOV     @#CONTRL,PCONTR ;SAVE CONTROL REGISTER FOR TYPE OUT
022362  012737  030060  001300        MOV     #30060,PMAINT   ;SAVE DATA IN MAINTENENCE REGISTER
022370  000002                        RTI                     ;RETURN TO TEST AND CHECK PARITY


                        ;;**********************************************************************
                        ;*TEST 44        PARITY REPORTING THRU THE MAP, CACHE DATA GROUP 1
                        ;*
```

```
                                        ;*      THIS TEST FORCES A PARITY ERROR IN CACHE GROUP 1 AT PHYSICAL
                                        ;*      ADDRESS 00040000 BY SETTING BIT 15 IN THE WORD.  THE TEST
                                        ;*      THEN FORCES GROUP 1 LOW BYTE PARITY BIT TO ZERO AND EVEN
                                        ;*      WORD HIGH BYTE PARITY BIT TO ONE BEFORE READING ADDRESS 00040000
                                        ;*      THRU THE UNIBUS MAP.  THEN IT CHECKS TO SEE THAT THE PARITY
                                        ;*      ABORT OCCURRED AND THAT THE CONDITION WAS CORRECT BEFORE
                                        ;*      GOING TO THE NEXT TEST.
                                        ;*
                                        ;*      ERRORS ARE REPORTED INLINE IN THE TEST CODE, AND THE PARITY
                                        ;*      VECTOR WILL TRAP TO LABEL 10$ AT THE END OF THIS TEST.
                                        ;*
                                        ;*      THIS TEST IS BEING RUN WITH ALL MEMORY REFERENCES GOING THRU
                                        ;*      THE UNIBUS MAP.
                                        ;*
                                        ;**************************************************************************
022372  000004                  TST44:  SCOPE
022374  012737  022660  001324          MOV     #SEOP,NXTTST    ;SET UP ESCAPE POINTER IN CASE OF
                                                                ;RANDOM PARITY ABORTS
022402  012737  177777  022546  20$:    MOV     #-1,21$         ;RESTORE NEG ONE FOR NEXT ITERATION
022410  032737  000010  177746          BIT     #BIT3,CONTRL    ;SEE IF GROUP 1 IS DISABLED
022416  001120                          BNE     $EOP            ;;BRANCH TO NEXT TEST IF GROUP 1 DISABLED
022420  013737  177746  001202          MOV     CONTRL,$TMP5    ;SAVE CONDITION OF CACHE
022426  012737  023604  001200          MOV     #23604,$TMP4    ;EXPECTED PARITY CONDITION GROUP 1
022434  005037  001274                  CLR     PPARER          ;CLEAR MEM ERROR REG STORAGE
022440  012737  177777  177744          MOV     #-1,@#MEMERR    ;CLEAR MEMORY ERROR REGISTER
022446  013737  001240  172354          MOV     LOWEST,KIPAR6   ;PUT UNIBUS ADDR OF MAP REG IN PAR6
022454  005077  156566                  CLR     @LREGU          ;CLEAR UPPER BITS ON MAP REG
022460  012777  040000  156556          MOV     #40000,@LREGL   ;LOAD LOWER 16 BITS OF MAP REG
022466  012705  177750                  MOV     #MAINT,R5       ;PUT ADDR OF MAINT REG IN R5
022472  012704  140000                  MOV     #140000,R4      ;SELECT PAR 6 BASE 0
022476  012714  100000                  MOV     #BIT15,(R4)     ;MAKE SURE I GET SOFT AND HARD ERRORS
                                                                ;BY LOADING 40000 WITH BIT 15, THIS
                                                                ;CAUSES THE HIGH BYTE P.B. TO BE ON
                                                                ;AND THE LOW BYTE P.B. TO BE OFF
022502  012737  000044  177746          MOV     #44,@#CONTRL    ;FORCE SELECTION OF GROUP 0 ON READ
022510  011401                          MOV     (R4),R1         ;PUT (40000) & (40002) IN GROUP 1
022512  005001                          CLR     R1              ;CLEAR R1 FOR MOMENT
022514  012737  022620  000114          MOV     #10$,CACHVEC    ;SET PARITY VECTOR TO 10$
022522  010137  177750          1$:     MOV     R1,@#MAINT      ;FORCE GP1 PARITY BITS TO 0 ON 2ND PASS
022526  000241                          CLC                     ;THIS IS A SYNC POINT FOR SCOPING
                                                                ;THE LOW BYTE HAS ITS P.B. OFF
022530  011401                          MOV     (R4),R1         ;DATA FETCH SHOULD CAUSE PARITY ABORT
022532  011503                          MOV     (R5),R3         ;SAVE MAINT REG IN CASE NO TRAP
022534  105015                          CLRB    (R5)            ;CLEAR MAINT REG IN CASE NO TRAP
022536  006701                          SXT     R1              ;DUMMY INST WITH P.B.'S OFF
022540  012701  020100                  MOV     #020100,R1      ;FORCE GROUP 1 LOW BYTE PARITY BIT
                                                                ;TO 0 AND MAIN MEMORY EVEN WORD HIGH
                                                                ;BYTE PARITY BIT TO 1
022544  005227                          INC     (PC)+           ;INCREMENT NEXT WORD TO 0
022546  177777          21$:    .WORD   -1                      ;
022550  001764                  BEQ     1$              ;BRANCH ON FIRST PASS ONLY
022552  023737  001200  001274          CMP     $TMP4,PPARER    ;SEE IF ERROR CONDITION MATCHES EXPECTED
022560  001402                          BEQ     2$              ;BRANCH IF ERROR WAS EXPECTED ONE
022562  010302                          MOV     R3,R2           ;R2 HOLDS MAINT REG FOR ERROR TYPE OUT
022564  104051                          ERROR   51              ;FAILURE IN PARITY REPORTING
022566  012737  177777  177744  2$:     MOV     #-1,MEMERR      ;CLEAR MEMORY ERROR REGISTER
```

```
                                                ;RESTORE PARITY SERVICE ROUTINE
                                                ;RESTORE CACHE TO FORMER CONDITION
                                                ;SET LOOP POINTER TO START OF TEST
                                                ;;TEST OVER GO TO NEXT TEST


                                                ;SAVE LOWER CACHE ADDR REG
                                                ;SAVE HIGH BITS OF FAILING ADDR
                                                ;SAVE MEMORY ERROR REGISTER
                                                ;SAVE CONTROL REGISTER FOR TYPE OUT
                                                ;SAVE DATA IN MAINTENENCE REGISTER
                                                ;RETURN TO TEST AND CHECK PARITY


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

                                .SBTTL  END OF PASS ROUTINE

                                ;*INCREMENT THE PASS NUMBER ($PASS)
                                ;*INDICATE END-OF-PROGRAM AFTER 1 PASSES THRU THE PROGRAM
                                ;*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

                                $EOP:
                                        SCOPE                   ;LOOP ON LAST TEST
                                        CLR     MMR0            ;TURN OFF FULL RELOCATION
                                        CLR     MMR3            ;DISABLE THE UNIBUS MAP
                                        TBITR                   ;RESTORE THE T BIT IF IT WAS ON
                                        CLR     $TSTNM          ;;ZERO THE TEST NUMBER
                                        CLR     $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
                                        INC     $PASS           ;;INCREMENT THE PASS NUMBER
                                        BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
                                        DEC     (PC)+           ;;LOOP?
                                $EOPCT: .WORD   1
                                        BGT     $DOAGN          ;;YES
                                        MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
                                $ENDCT: .WORD   1
                                        $EOPCT
                                        TYPE    ,65$            ;;TYPE ASCIZ STRING
                                        BR      64$             ;;GET OVER THE ASCIZ
                                ;;65$:  .ASCIZ  <12><15>/END PASS #/
                                64$:
                                        MOV     $PASS,-(SP)     ;;SAVE $PASS FOR TYPEOUT
                                                                ;;TYPE PASS NUMBER
                                        TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
                                        TYPE    ,67$            ;;TYPE ASCIZ STRING
                                        BR      66$             ;;GET OVER THE ASCIZ
                                ;;67$:  .ASCIZ  /  TOTAL ERRORS SINCE LAST REPORT /
                                66$:
                                        MOV     $ERTTL,-(SP)    ;;SAVE $ERTTL FOR TYPEOUT
                                                                ;;TOTAL NUMBER OF ERRORS
                                        TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
```

```
 4794  023042  104400  001215                  TYPE    ,$CRLF          ;;TYPE CARRIAGE RETURN, LINE FEED
 4795  023046  005037  001112                  CLR     $ERTTL          ;;CLEAR ERROR TOTAL
 4796  023052  013700  000042          $GET42: MOV     @#42,R0         ;;GET MONITOR ADDRESS
 4797  023056  001414                          BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
 4798  023060  005046                          CLR     -(SP)           ;;INSURE THE "T" BIT IS CLEAR
 4799  023062  012746  023070                  MOV     #$CLR.T,-(SP)   ;;SETUP FOR AN RTI OR RTT
 4800  023066  000427                          BR      $RTRN           ;;GO DO AN RTI OR RTT TO LOAD THE PSW
 4801                                                                  ;;WITH A CLEARED "T" BIT
 4802  023070                          $CLR.T:
 4803  023070  013700  000042                  MOV     @#42,R0         ;;INSURE R0 CONTAINS THE MONITORS
 4804  023074  001405                          BEQ     $DOAGN          ;;RETURN ADDRESS
 4805  023076  000005                          RESET                   ;;CLEAR THE WORLD
 4806  023100  004710                  $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
 4807  023102  000240                          NOP                     ;;SAVE ROOM
 4808  023104  000240                          NOP                     ;;FOR
 4809  023106  000240                          NOP                     ;;ACT11
 4810  023110                          $DOAGN:
 4811  023110  013746  177776                  MOV     @#PS,-(SP)      ;;PUT THE PS ON THE STACK AND
 4812  023114  042716  000020                  BIC     #20,(SP)        ;;CLEAR THE "T" BIT
 4813  023120  032737  010000  177570          BIT     #BIT12,@#SWR    ;;RUN WITH TRACE TRAP?
 4814  023126  001005                          BNE     1$              ;;BR IF NO
 4815  023130  005137  023154                  COM     $TBIT           ;;IS IT TIME FOR TRACE TRAP
 4816  023134  100402                          BMI     1$              ;;BR IF NO
 4817  023136  052716  000020                  BIS     #20,(SP)        ;;SET TRACE TRAP
 4818  023142  012746  023150          1$:     MOV     #$LOOP,-(SP)    ;;JUMP TO START OF TEST
 4819  023146  000002                  $RTRN:  RTI                     ;;RETURN--THIS IS CHANGED TO
 4820                                                                  ;;AN "RTT" IF "RTT" IS A LEGAL
 4821                                                                  ;;INSTRUCTION
 4822  023150                          $LOOP:
 4823  023150  000137  010346                  JMP     @#LOOP          ;;RETURN
 4824  023154  000000                  $TBIT:  .WORD   0               ;;"T" BIT STATE INDICATOR
 4825  023156     377     377     000  $ENULL: .BYTE   -1,-1,0         ;;NULL CHARACTER STRING
 4826          023162                          .EVEN
 4827
 4828
 4829                                  .SBTTL  ERROR MESSAGES AND DATA TABLES
 4830  023162  047516  020124  044124  EM1:    .ASCIZ  ?NOT THE CORRECT TRAP CONDITION THRU ERRVEC (#004)?
 4831  023170  020105  047503  051122
 4832  023176  041505  020124  051124
 4833  023204  050101  041440  047117
 4834  023212  044504  044524  047117
 4835  023220  052040  051110  020125
 4836  023226  051105  053122  041505
 4837  023234  024040  030043  032060
 4838  023242  000051
 4839  023244  047125  054105  042520  EM2:    .ASCIZ  ?UNEXPECTED CPU TRAP THRU ERRVEC (#004)?
 4840  023252  052103  042105  041440
 4841  023260  052520  052040  040522
 4842  023266  020120  044124  052522
 4843  023274  042440  051122  042526
 4844  023302  020103  021450  030060
 4845  023310  024464     000
 4846  023313     125  042516  050130  EM3:    .ASCIZ  ?UNEXPECTED CACHE PARITY ERROR THRU CACHVEC, WILL RETRY TEST ONCE?
 4847  023320  041505  042524  020104
 4848  023326  040503  044103  020105
 4849  023334  040520  044522  054524
```

```
 4850  023342  042440  051122  051117
 4851  023350  052040  051110  020125
 4852  023356  040503  044103  042526
 4853  023364  026103  053440  046111
 4854  023372  020114  042522  051124
 4855  023400  020131  042524  052123
 4856  023406  047440  041516  000105
 4857  023414  047125  054105  042520  EM4:    .ASCIZ  ?UNEXPECTED MAIN MEMORY PARITY ERROR THRU CACHVEC, WILL RETRY TEST ONCE?
 4858  023422  052103  042105  046440
 4859  023430  044501  020116  042515
 4860  023436  047515  054522  050040
 4861  023444  051101  052111  020131
 4862  023452  051105  047522  020122
 4863  023460  044124  052522  041440
 4864  023466  041501  053110  041505
 4865  023474  020054  044527  046114
 4866  023502  051040  052105  054522
 4867  023510  052040  051505  020124
 4868  023516  047117  042503     000
 4869  023523     125  042516  050130  EM5:    .ASCIZ  ?UNEXPECTED MEMORY MANAGEMENT TRAP, MEMORY MANAGEMENT STATUS REGISTERS?
 4870  023530  041505  042524  020104
 4871  023536  042515  047515  054522
 4872  023544  046440  047101  043501
 4873  023552  046505  047105  020124
 4874  023560  051124  050101  020054
 4875  023566  042515  047515  054522
 4876  023574  046440  047101  043501
 4877  023602  046505  047105  020124
 4878  023610  052123  052101  051525
 4879  023616  051040  043505  051511
 4880  023624  042524  051522     000
 4881  023631     123  046525  040515  EM6:    .ASCIZ  ?SUMMARY OF MAP REGISTERS THAT TIMED OUT ON READ?
 4882  023636  054522  047440  020106
 4883  023644  040515  020120  042522
 4884  023652  044507  052123  051105
 4885  023660  020123  044124  052101
 4886  023666  052040  046511  042105
 4887  023674  047440  052125  047440
 4888  023702  020116  042522  042101
 4889  023710     000
 4890  023711     123  046525  040515  EM7:    .ASCIZ  ?SUMMARY OF CACHE REGISTERS THAT TIMED OUT ON READ?
 4891  023716  054522  047440  020106
 4892  023724  040503  044103  020105
 4893  023732  042522  044507  052123
 4894  023740  051105  020123  044124
 4895  023746  052101  052040  046511
 4896  023754  042105  047440  052125
 4897  023762  047440  020116  042522
 4898  023770  042101     000
 4899  023773     123  046525  040515  EM10:   .ASCIZ  ?SUMMARY OF MAP REGISTERS NOT HOLDING ZERO IN LOW 16 BITS?
 4900  024000  054522  047440  020106
 4901  024006  040515  020120  042522
 4902  024014  044507  052123  051105
 4903  024022  020123  047516  020124
 4904  024030  047510  042114  047111
 4905  024036  020107  042532  047522
```

```
4906  024044  044440  020116  047514
4907  024052  020127  033061  041040
4908  024060  052111  000123
4909  024064  052523  046515  051101  EM11:   .ASCIZ  ?SUMMARY OF MAP REGISTERS NOT HOLDING ZERO IN UPPER 6 BITS?
4910  024072  020131  043117  046440
4911  024100  050101  051040  043505
4912  024106  051511  042524  051522
4913  024114  047040  052117  044040
4914  024122  046117  044504  043516
4915  024130  055040  051105  020117
4916  024136  047111  052440  050120
4917  024144  051105  033040  041040
4918  024152  052111  000123
4919  024156  047520  051523  041111  EM12:   .ASCIZ  ?POSSIBLE ERROR IN MAP REGISTER DATA PATH (MAP REG 00)?
4920  024164  042514  042440  051122
4921  024172  051117  044440  020116
4922  024200  040515  020120  042522
4923  024206  044507  052123  051105
4924  024214  042040  052101  020101
4925  024222  040520  044124  024040
4926  024230  040515  020120  042522
4927  024236  020107  030060  000051
4928  024244  047516  020127  051120  EM13:   .ASCIZ  ?NOW PROBABLE ERROR IN MAP REGISTER DATA PATH (MAP REG 20)?
4929  024252  041117  041101  042514
4930  024260  042440  051122  051117
4931  024266  044440  020116  040515
4932  024274  020120  042522  044507
4933  024302  052123  051105  042040
4934  024310  052101  020101  040520
4935  024316  044124  024040  040515
4936  024324  020120  042522  020107
4937  024332  030062  000051
4938  024336  052523  046515  051101  EM14:   .ASCIZ  ?SUMMARY OF DUAL ADDRESSING ERRORS ON LOADING MAP REGISTERS?
4939  024344  020131  043117  042040
4940  024352  040525  020114  042101
4941  024360  051104  051505  044523
4942  024366  043516  042440  051122
4943  024374  051117  020123  047117
4944  024402  046040  040517  044504
4945  024410  043516  046440  050101
4946  024416  051040  043505  051511
4947  024424  042524  051522     000
4948  024431     123  046525  040515  EM15:   .ASCIZ  ?SUMMARY OF COUNT PATTERN FAILURES IN LOWER 16 BITS OF MAP REGISTERS?
4949  024436  054522  047440  020106
4950  024444  047503  047125  020124
4951  024452  040520  052124  051105
4952  024460  020116  040506  046111
4953  024466  051125  051505  044440
4954  024474  020116  047514  042527
4955  024502  020122  033061  041040
4956  024510  052111  020123  043117
4957  024516  046440  050101  051040
4958  024524  043505  051511  042524
4959  024532  051522     000
4960  024535     123  046525  040515  EM16:   .ASCIZ  ?SUMMARY OF COUNT PATTERN FAILURES IN UPPER 6 BITS OF MAP REGISTERS?
4961  024542  054522  047440  020106
```

```
 4962  024550  047503  047125  020124
 4963  024556  040520  052124  051105
 4964  024564  020116  040506  046111
 4965  024572  051125  051505  044440
 4966  024600  020116  050125  042520
 4967  024606  020122  020066  044502
 4968  024614  051524  047440  020106
 4969  024622  040515  020120  042522
 4970  024630  044507  052123  051105
 4971  024636  000123
 4972  024640  047503  046125  020104  EM17:   .ASCII  ?COULD NOT CLEAR CACHE CONTROL REGISTER?<CRLF>
 4973  024646  047516  020124  046103
 4974  024654  040505  020122  040503
 4975  024662  044103  020105  047503
 4976  024670  052116  047522  020114
 4977  024676  042522  044507  052123
 4978  024704  051105     200
 4979  024707     120  051517  044523          .ASCIZ  ?POSSIBLE ERROR IN CACHE REGISTER DATA PATH?
 4980  024714  046102  020105  051105
 4981  024722  047522  020122  047111
 4982  024730  041440  041501  042510
 4983  024736  051040  043505  051511
 4984  024744  042524  020122  040504
 4985  024752  040524  050040  052101
 4986  024760  000110
 4987  024762  047503  046125  020104  EM20:   .ASCII  ?COULD NOT CLEAR CACHE MAINTENENCE REGISTER?<CRLF>
 4988  024770  047516  020124  046103
 4989  024776  040505  020122  040503
 4990  025004  044103  020105  040515
 4991  025012  047111  042524  042516
 4992  025020  041516  020105  042522
 4993  025026  044507  052123  051105
 4994  025034     200
 4995  025035     120  051517  044523          .ASCIZ  ?POSSIBLE ERROR IN CACHE REGISTER DATA PATH?
 4996  025042  046102  020105  051105
 4997  025050  047522  020122  047111
 4998  025056  041440  041501  042510
 4999  025064  051040  043505  051511
 5000  025072  042524  020122  040504
 5001  025100  040524  050040  052101
 5002  025106  000110
 5003  025110  047503  046125  020104  EM21:   .ASCII  ?COULD NOT READ 177740 FROM CACHE LO ADDRESS REG (LOADRS)?<CRLF>
 5004  025116  047516  020124  042522
 5005  025124  042101  030440  033467
 5006  025132  032067  020060  051106
 5007  025140  046517  041440  041501
 5008  025146  042510  046040  020117
 5009  025154  042101  051104  051505
 5010  025162  020123  042522  020107
 5011  025170  046050  040517  051104
 5012  025176  024523     200
 5013  025201     120  051517  044523          .ASCIZ  ?POSSIBLE ERROR IN CACHE REGISTER DATA PATH?
 5014  025206  046102  020105  051105
 5015  025214  047522  020122  047111
 5016  025222  041440  041501  042510
 5017  025230  051040  043505  051511
```

```
5018  025236  042524  020122  040504
5019  025244  040524  050040  052101
5020  025252  000110
5021  025254  047503  046125  020104  EM22:   .ASCII  ?COULD NOT READ 000003 FROM CACHE HI ADDRESS REG (HIADRS)?<CRLF>
5022  025262  047516  020124  042522
5023  025270  042101  030040  030060
5024  025276  030060  020063  051106
5025  025304  046517  041440  041501
5026  025312  042510  044040  020111
5027  025320  042101  051104  051505
5028  025326  020123  042522  020107
5029  025334  044050  040511  051104
5030  025342  024523     200
5031  025345     120  051517  044523          .ASCIZ  ?POSSIBLE ERROR IN CACHE REGISTER DATA PATH?
5032  025352  046102  020105  051105
5033  025360  047522  020122  047111
5034  025366  041440  041501  042510
5035  025374  051040  043505  051511
5036  025402  042524  020122  040504
5037  025410  040524  050040  052101
5038  025416  000110
5039  025420  052523  046515  051101  EM23:   .ASCIZ  ?SUMMARY OF COUNT PATTERN FAILURES IN CACHE CONTROL REGISTER?
5040  025426  020131  043117  041440
5041  025434  052517  052116  050040
5042  025442  052101  042524  047122
5043  025450  043040  044501  052514
5044  025456  042522  020123  047111
5045  025464  041440  041501  042510
5046  025472  041440  047117  051124
5047  025500  046117  051040  043505
5048  025506  051511  042524  000122
5049  025514  052523  046515  051101  EM24:   .ASCIZ  ?SUMMARY OF COUNT PATTERN FAILURES IN CACHE MAINTENENCE REGISTER?
5050  025522  020131  043117  041440
5051  025530  052517  052116  050040
5052  025536  052101  042524  047122
5053  025544  043040  044501  052514
5054  025552  042522  020123  047111
5055  025560  041440  041501  042510
5056  025566  046440  044501  052116
5057  025574  047105  047105  042503
5058  025602  051040  043505  051511
5059  025610  042524  000122
5060  025614  042522  042506  042522  EM25:   .ASCIZ  ?REFERENCED MAP REGISTER 0 WITH ADDRESS ONE BIT DIFFERENT THAN 770200?
5061  025622  041516  042105  046440
5062  025630  050101  051040  043505
5063  025636  051511  042524  020122
5064  025644  020060  044527  044124
5065  025652  040440  042104  042522
5066  025660  051523  047440  042516
5067  025666  041040  052111  042040
5068  025674  043111  042506  042522
5069  025702  052116  052040  040510
5070  025710  020116  033467  031060
5071  025716  030060     000
5072  025721     122  043105  051105  EM26:   .ASCII  ?REFERENCED CACHE LOW ADDRESS REGISTER WITH ADDRESS ONE BIT?<CRLF>
5073  025726  047105  042503  020104
```

```
 5074  025734  040503  044103  020105
 5075  025742  047514  020127  042101
 5076  025750  051104  051505  020123
 5077  025756  042522  044507  052123
 5078  025764  051105  053440  052111
 5079  025772  020110  042101  051104
 5080  026000  051505  020123  047117
 5081  026006  020105  044502  100124
 5082  026014  044504  043106  051105          .ASCIZ  ?DIFFERENT THAN 777740?
 5083  026022  047105  020124  044124
 5084  026030  047101  033440  033467
 5085  026036  032067  000060
 5086  026042  040503  023516  020124  EM30:   .ASCII  ?CAN'T GET TO MAIN MEMORY FROM UNIBUS WITH THE MAP OFF?<CRLF>
 5087  026050  042507  020124  047524
 5088  026056  046440  044501  020116
 5089  026064  042515  047515  054522
 5090  026072  043040  047522  020115
 5091  026100  047125  041111  051525
 5092  026106  053440  052111  020110
 5093  026114  044124  020105  040515
 5094  026122  020120  043117  100106
 5095  026130  047523  044440  046047          .ASCIZ  ?SO I'LL JUMP TO THE SIZE JUMPER TEST FOR VERIFICATION?
 5096  026136  020114  052512  050115
 5097  026144  052040  020117  044124
 5098  026152  020105  044523  042532
 5099  026160  045040  046525  042520
 5100  026166  020122  042524  052123
 5101  026174  043040  051117  053040
 5102  026202  051105  043111  041511
 5103  026210  052101  047511  000116
 5104  026216  052523  046515  051101  EM31:   .ASCIZ  ?SUMMARY OF COUNT PATTERN FAILURES ON THE UNIBUS DATA PATH?
 5105  026224  020131  043117  041440
 5106  026232  052517  052116  050040
 5107  026240  052101  042524  047122
 5108  026246  043040  044501  052514
 5109  026254  042522  020123  047117
 5110  026262  052040  042510  052440
 5111  026270  044516  052502  020123
 5112  026276  040504  040524  050040
 5113  026304  052101  000110
 5114  026310  047125  041111  051525  EM32:   .ASCIZ  ?UNIBUS MAP IS RELOCATING WHEN NOT ENABLED?
 5115  026316  046440  050101  044440
 5116  026324  020123  042522  047514
 5117  026332  040503  044524  043516
 5118  026340  053440  042510  020116
 5119  026346  047516  020124  047105
 5120  026354  041101  042514  000104
 5121  026362  040503  047116  052117  EM33:   .ASCII  ?CANNOT USE ANY OF THE MAP REGISTERS, OR PHYSICAL?<CRLF>
 5122  026370  052440  042523  040440
 5123  026376  054516  047440  020106
 5124  026404  044124  020105  040515
 5125  026412  020120  042522  044507
 5126  026420  052123  051105  026123
 5127  026426  047440  020122  044120
 5128  026434  051531  041511  046101
 5129  026442     200
```

```
5130  026443    101  042104  042522          .ASCIZ  ?ADDRESS BIT14 IS STUCK LOW. MUST RESTART PROGRAM IF NO LOOP?
5131  026450 051523  041040  052111
5132  026456 032061  044440  020123
5133  026464 052123  041525  020113
5134  026472 047514  027127  046440
5135  026500 051525  020124  042522
5136  026506 052123  051101  020124
5137  026514 051120  043517  040522
5138  026522 020115  043111  047040
5139  026530 020117  047514  050117
5140  026536    000
5141  026537    124  042510  047040  EM34:   .ASCII  ?THE NUMBER OF MAP REGISTERS REMOVED BY JUMPER SETTING DOES?<CRLF>
5142  026544 046525  042502  020122
5143  026552 043117  046440  050101
5144  026560 051040  043505  051511
5145  026566 042524  051522  051040
5146  026574 046505  053117  042105
5147  026602 041040  020131  052512
5148  026610 050115  051105  051440
5149  026616 052105  044524  043516
5150  026624 042040  042517  100123
5151  026632 047516  020124  043501          .ASCIZ  ?NOT AGREE WITH THE NUMBER FOUND TO BE MISSING?
5152  026640 042522  020105  044527
5153  026646 044124  052040  042510
5154  026654 047040  046525  042502
5155  026662 020122  047506  047125
5156  026670 020104  047524  041040
5157  026676 020105  044515  051523
5158  026704 047111  000107
5159  026710 044124  020105  044523  EM35:   .ASCII  ?THE SIZE JUMPERS ON THE UNIBUS MAP ARE NOT SET IN?<CRLF>
5160  026716 042532  045040  046525
5161  026724 042520  051522  047440
5162  026732 020116  044124  020105
5163  026740 047125  041111  051525
5164  026746 046440  050101  040440
5165  026754 042522  047040  052117
5166  026762 051440  052105  044440
5167  026770 100116
5168  026772 044124  044505  020122          .ASCII  ?THEIR DEFAULT POSITION, WHICH ALLOWS UNIBUS ADDRESSES?<CRLF>
5169  027000 042504  040506  046125
5170  027006 020124  047520  044523
5171  027014 044524  047117  020054
5172  027022 044127  041511  020110
5173  027030 046101  047514  051527
5174  027036 052440  044516  052502
5175  027044 020123  042101  051104
5176  027052 051505  042523  100123
5177  027060 030060  030060  030060          .ASCII  ?000000 TO 757776 TO REFERENCE MAIN MEMORY?<CRLF>
5178  027066 052040  020117  032467
5179  027074 033467  033067  052040
5180  027102 020117  042522  042506
5181  027110 042522  041516  020105
5182  027116 040515  047111  046440
5183  027124 046505  051117  100131
5184  027132 044124  044505  020122          .ASCIZ  ?THEIR CURRENT SETTING ALLOWS ONLY:?
5185  027140 052503  051122  047105
```

```
5186  027146  020124  042523  052124
5187  027154  047111  020107  046101
5188  027162  047514  051527  047440
5189  027170  046116  035131     000
5190  027175     115  050101  051040  EM36:   .ASCIZ  ?MAP REGISTER UNDER TEST DID NOT RESPOND IN DUAL MAPPING TEST?
5191  027202  043505  051511  042524
5192  027210  020122  047125  042504
5193  027216  020122  042524  052123
5194  027224  042040  042111  047040
5195  027232  052117  051040  051505
5196  027240  047520  042116  044440
5197  027246  020116  052504  046101
5198  027254  046440  050101  044520
5199  027262  043516  052040  051505
5200  027270  000124
5201  027272  052523  046515  051101  EM37:   .ASCIZ  ?SUMMARY OF UNIBUS ADDRESS ERRORS, WITH THE MAP RELOCATION DISABLED?
5202  027300  020131  043117  052440
5203  027306  044516  052502  020123
5204  027314  042101  051104  051505
5205  027322  020123  051105  047522
5206  027330  051522  020054  044527
5207  027336  044124  052040  042510
5208  027344  046440  050101  051040
5209  027352  046105  041517  052101
5210  027360  047511  020116  044504
5211  027366  040523  046102  042105
5212  027374     000
5213  027375     115  044501  020116  EM40:   .ASCIZ  ?MAIN MEMORY TIMEOUT OVER THE UNIBUS DID NOT OCCUR PROPERLY?
5214  027402  042515  047515  054522
5215  027410  052040  046511  047505
5216  027416  052125  047440  042526
5217  027424  020122  044124  020105
5218  027432  047125  041111  051525
5219  027440  042040  042111  047040
5220  027446  052117  047440  041503
5221  027454  051125  050040  047522
5222  027462  042520  046122  000131
5223  027470  042522  047514  040503  EM41:   .ASCIZ  ?RELOCATION THRU THE MAP WAS NOT CORRECT, FULL ADD?
5224  027476  044524  047117  052040
5225  027504  051110  020125  044124
5226  027512  020105  040515  020120
5227  027520  040527  020123  047516
5228  027526  020124  047503  051122
5229  027534  041505  026124  043040
5230  027542  046125  020114  042101
5231  027550  000104
5232  027552  042522  047514  040503  EM42:   .ASCIZ  ?RELOCATION THRU THE MAP WAS NOT CORRECT, CARRY PROPAGATION?
5233  027560  044524  047117  052040
5234  027566  051110  020125  044124
5235  027574  020105  040515  020120
5236  027602  040527  020123  047516
5237  027610  020124  047503  051122
5238  027616  041505  026124  041440
5239  027624  051101  054522  050040
5240  027632  047522  040520  040507
5241  027640  044524  047117     000
```

```
5242  027645    124  042510  052040  EM43:   .ASCIZ  ?THE TOP OF MEMORY IS DIFFERENT THAN THE SIZE JUMPERS?
5243  027652 050117  047440  020106
5244  027660 042515  047515  054522
5245  027666 044440  020123  044504
5246  027674 043106  051105  047105
5247  027702 020124  044124  047101
5248  027710 052040  042510  051440
5249  027716 055111  020105  052512
5250  027724 050115  051105  000123
5251  027732 040520  044522  054524  EM44:   .ASCIZ  ?PARITY REPORTING THRU THE MAP IS NOT CORRECT?
5252  027740 051040  050105  051117
5253  027746 044524  043516  052040
5254  027754 051110  020125  044124
5255  027762 020105  040515  020120
5256  027770 051511  047040  052117
5257  027776 041440  051117  042522
5258  030004 052103     000
5259  030007    115  044501  020116  EM45:   .ASCII  ?MAIN MEMORY TIMEOUT OVER THE UNIBUS DID NOT OCCUR PROPERLY?<CRLF>
5260  030014 042515  047515  054522
5261  030022 052040  046511  047505
5262  030030 052125  047440  042526
5263  030036 020122  044124  020105
5264  030044 047125  041111  051525
5265  030052 042040  042111  047040
5266  030060 052117  047440  041503
5267  030066 051125  050040  047522
5268  030074 042520  046122  100131
5269  030102 042524  052123  041440          .ASCIZ  ?TEST CODE BEING RUN OVER UNIBUS?
5270  030110 042117  020105  042502
5271  030116 047111  020107  052522
5272  030124 020116  053117  051105
5273  030132 052440  044516  052502
5274  030140 000123
5275  030142 042522  047514  040503  EM46:   .ASCII  ?RELOCATION THRU THE MAP WAS NOT CORRECT, FULL ADD?<CRLF>
5276  030150 044524  047117  052040
5277  030156 051110  020125  044124
5278  030164 020105  040515  020120
5279  030172 040527  020123  047516
5280  030200 020124  047503  051122
5281  030206 041505  026124  043040
5282  030214 046125  020114  042101
5283  030222 100104
5284  030224 042524  052123  041440          .ASCIZ  ?TEST CODE BEING RUN OVER UNIBUS?
5285  030232 042117  020105  042502
5286  030240 047111  020107  052522
5287  030246 020116  053117  051105
5288  030254 052440  044516  052502
5289  030262 000123
5290  030264 042522  047514  040503  EM47:   .ASCII  ?RELOCATION THRU THE MAP WAS NOT CORRECT, CARRY PROPAGATION?<CRLF>
5291  030272 044524  047117  052040
5292  030300 051110  020125  044124
5293  030306 020105  040515  020120
5294  030314 040527  020123  047516
5295  030322 020124  047503  051122
5296  030330 041505  026124  041440
5297  030336 051101  054522  050040
```

```
5298  030344  047522  040520  040507
5299  030352  044524  047117     200
5300  030357     124  051505  020124          .ASCIZ  ?TEST CODE BEING RUN OVER UNIBUS?
5301  030364  047503  042504  041040
5302  030372  044505  043516  051040
5303  030400  047125  047440  042526
5304  030406  020122  047125  041111
5305  030414  051525     000
5306  030417     124  042510  052040  EM50:   .ASCII  ?THE TOP OF MEMORY IS DIFFERENT THAN THE SIZE JUMPERS?<CRLF>
5307  030424  050117  047440  020106
5308  030432  042515  047515  054522
5309  030440  044440  020123  044504
5310  030446  043106  051105  047105
5311  030454  020124  044124  047101
5312  030462  052040  042510  051440
5313  030470  055111  020105  052512
5314  030476  050115  051105  100123
5315  030504  042524  052123  041440          .ASCIZ  ?TEST CODE BEING RUN OVER UNIBUS?
5316  030512  042117  020105  042502
5317  030520  047111  020107  052522
5318  030526  020116  053117  051105
5319  030534  052440  044516  052502
5320  030542  000123
5321  030544  040520  044522  054524  EM51:   .ASCII  ?PARITY REPORTING THRU THE MAP IS NOT CORRECT?<CRLF>
5322  030552  051040  050105  051117
5323  030560  044524  043516  052040
5324  030566  051110  020125  044124
5325  030574  020105  040515  020120
5326  030602  051511  047040  052117
5327  030610  041440  051117  042522
5328  030616  052103     200
5329  030621     124  051505  020124          .ASCIZ  ?TEST CODE BEING RUN OVER UNIBUS?
5330  030626  047503  042504  041040
5331  030634  044505  043516  051040
5332  030642  047125  047440  042526
5333  030F50  020122  047125  041111
5334  030656  051525     000
5335  030661     123  046525  040515  EM52:   .ASCIZ  ?SUMMARY OF DUAL MAPPING ERRORS?
5336  030666  054522  047440  020106
5337  030674  052504  046101  046440
5338  030702  050101  044520  043516
5339  030710  042440  051122  051117
5340  030716  000123
5341  030720  044124  020105  047506  EM201:  .ASCIZ  ?THE FOLLOWING REGISTERS TIMED OUT WHEN READ?
5342  030726  046114  053517  047111
5343  030734  020107  042522  044507
5344  030742  052123  051105  020123
5345  030750  044524  042515  020104
5346  030756  052517  020124  044127
5347  030764  047105  051040  040505
5348  030772  000104
5349  030774  044124  020105  047506  EM202:  .ASCIZ  ?THE FOLLOWING MAP REGISTERS WILL NOT CLEAR?
5350  031002  046114  053517  047111
5351  031010  020107  040515  020120
5352  031016  042522  044507  052123
5353  031024  051105  020123  044527
```

```
5354  031032  046114  047040  052117
5355  031040  041440  042514  051101
5356  031046     000
5357  031047     124  042510  043040  EM203:  .ASCIZ  ?THE FOLLOWING ARE DUAL ADDRESSING ERRORS IN THE UNIBUS MAP?
5358  031054  046117  047514  044527
5359  031062  043516  040440  042522
5360  031070  042040  040525  020114
5361  031076  042101  051104  051505
5362  031104  044523  043516  042440
5363  031112  051122  051117  020123
5364  031120  047111  052040  042510
5365  031126  052440  044516  052502
5366  031134  020123  040515  000120
5367  031142  044124  020105  047503  EM204:  .ASCIZ  ?THE COUNT PATTERN THRU THE MAP REGISTERS FAILED?
5368  031150  047125  020124  040520
5369  031156  052124  051105  020116
5370  031164  044124  052522  052040
5371  031172  042510  046440  050101
5372  031200  051040  043505  051511
5373  031206  042524  051522  043040
5374  031214  044501  042514  000104
5375  031222  047125  041111  051525  EM205:  .ASCIZ  ?UNIBUS DATA PATH COUNT PATTERN FAILURE?
5376  031230  042040  052101  020101
5377  031236  040520  044124  041440
5378  031244  052517  052116  050040
5379  031252  052101  042524  047122
5380  031260  043040  044501  052514
5381  031266  042522     000
5382  031271     125  044516  052502  EM206:  .ASCIZ  ?UNIBUS ADDRESSING ERRORS, MAP RELOCATION DISABLED?
5383  031276  020123  042101  051104
5384  031304  051505  044523  043516
5385  031312  042440  051122  051117
5386  031320  026123  046440  050101
5387  031326  051040  046105  041517
5388  031334  052101  047511  020116
5389  031342  044504  040523  046102
5390  031350  042105     000
5391  031353     103  052517  052116  EM207:  .ASCIZ  ?COUNT PATTERN FAILURES IN CACHE REGISTERS?
5392  031360  050040  052101  042524
5393  031366  047122  043040  044501
5394  031374  052514  042522  020123
5395  031402  047111  041440  041501
5396  031410  042510  051040  043505
5397  031416  051511  042524  051522
5398  031424     000
5399
5400  031425     122  041505  044505  DH1:    .ASCIZ  ?RECEIVD EXPECTD TESTNO  PC AT ABORT?
5401  031432  042126  042440  050130
5402  031440  041505  042124  052040
5403  031446  051505  047124  020117
5404  031454  050040  020103  052101
5405  031462  040440  047502  052122
5406  031470     000
5407  031471     122  041505  044505  DH2:    .ASCIZ  ?RECEIVD TESTNO  PC AT ABORT?
5408  031476  042126  052040  051505
5409  031504  047124  020117  050040
```

```
5410  031512  020103  052101  040440
5411  031520  047502  052122     000
5412  031536     103  047117  044504  DH3:    .ASCII  ?CONDITN ADDRESS   MAINTEN CONTROL?<CRLF>
5413  031532  047124  040440  042104
5414  031540  042522  051523  020040
5415  031546  046440  044501  052116
5416  031554  047105  041440  047117
5417  031562  051124  046117     200
5418  031567     122  041505  044505          .ASCIZ  ?RECEIVD REFERENCD REGISTR REGISTR TESTNO  PC AT ABORT?
5419  031574  042126  051040  043105
5420  031602  051105  047105  042103
5421  031610  051040  043505  051511
5422  031616  051124  051040  043505
5423  031624  051511  051124  052040
5424  031632  051505  047124  020117
5425  031640  050040  020103  052101
5426  031646  040440  047502  052122
5427  031654     000
5428  031655     123  040524  052524  DH5:    .ASCII  ?STATUS  AUTOI/D VIRTUAL?<CRLF>
5429  031662  020123  040440  052125
5430  031670  044517  042057  053040
5431  031676  051111  052524  046101
5432  031704     200
5433  031705     122  043505  051511          .ASCIZ  ?REGISTR REGISTR ADDRESS TESTNO  PC AT ABORT?
5434  031712  051124  051040  043505
5435  031720  051511  051124  040440
5436  031726  042104  042522  051523
5437  031734  052040  051505  047124
5438  031742  020117  050040  020103
5439  031750  052101  040440  047502
5440  031756  052122     000
5441  031761     122  043505  042101  DH6:    .ASCII  ?REGADRS REGADRS?<CRLF>
5442  031766  051522  051040  043505
5443  031774  042101  051522     200
5444  032001     040  047442  021122          .ASCIZ  ? "OR"    "AND"  #ERRORS TESTNO  ERRORPC?
5445  032006  020040  020040  040442
5446  032014  042116  020042  021440
5447  032022  051105  047522  051522
5448  032030  052040  051505  047124
5449  032036  020117  042440  051122
5450  032044  051117  041520     000
5451  032051     122  043505  042101  DH10:   .ASCII  ?REGADRS REGADRS RECEIVD RECEIVD?<CRLF>
5452  032056  051522  051040  043505
5453  032064  042101  051522  051040
5454  032072  041505  044505  042126
5455  032100  051040  041505  044505
5456  032106  042126     200
5457  032111     040  047442  021122          .ASCIZ  ? "OR"    "AND"   "OR"    "AND"  #ERRORS TESTNO  ERRORPC?
5458  032116  020040  020040  040442
5459  032124  042116  020042  020040
5460  032132  047442  021122  020040
5461  032140  020040  040442  042116
5462  032146  020042  021440  051105
5463  032154  047522  051522  052040
5464  032162  051505  047124  020117
5465  032170  042440  051122  051117
```

```
5466  032176  041520     000
5467  032201     103  052517  052116  DH12:   .ASCII  ?COUNT   COUNT?<CRLF>
5468  032206  020040  041440  052517
5469  032214  052116     200
5470  032217     105  050130  041505          .ASCIZ  ?EXPECTD RECEIVD TESTNO  ERRORPC?
5471  032224  042124  051040  041505
5472  032232  044505  042126  052040
5473  032240  051505  047124  020117
5474  032246  042440  051122  051117
5475  032254  041520     000
5476  032257     122  043505  047514  DH14:   .ASCII  ?REGLOAD REGLOAD REGDUAL REGDUAL?<CRLF>
5477  032264  042101  051040  043505
5478  032272  047514  042101  051040
5479  032300  043505  052504  046101
5480  032306  051040  043505  052504
5481  032314  046101     200
5482  032317     040  047442  021122          .ASCIZ  ? "OR"    "AND"   "OR"    "AND"  #ERRORS TESTNO?
5483  032324  020040  020040  040442
5484  032332  042116  020042  020040
5485  032340  047442  021122  020040
5486  032346  020040  040442  042116
5487  032354  020042  021440  051105
5488  032362  047522  051522  052040
5489  032370  051505  047124  000117
5490  032376  040515  051120  043505  DH15:   .ASCII  ?MAPREG  MAPREG  EXPECTD EXPECTD RECEIVD RECEIVD?<CRLF>
5491  032404  020040  040515  051120
5492  032412  043505  020040  054105
5493  032420  042520  052103  020104
5494  032426  054105  042520  052103
5495  032434  020104  042522  042503
5496  032442  053111  020104  042522
5497  032450  042503  053111  100104
5498  032456  021040  051117  020042          .ASCIZ  ? "OR"    "AND"   "OR"    "AND"   "OR"    "AND"  #ERRORS TESTNO?
5499  032464  020040  021040  047101
5500  032472  021104  020040  021040
5501  032500  051117  020042  020040
5502  032506  021040  047101  021104
5503  032514  020040  021040  051117
5504  032522  020042  020040  021040
5505  032530  047101  021104  020040
5506  032536  042443  051122  051117
5507  032544  020123  042524  052123
5508  032552  047516     000
5509  032555     122  041505  044505  DH17:   .ASCIZ  ?RECEIVD TESTNO  ERRORPC?
5510  032562  042126  052040  051505
5511  032570  047124  020117  042440
5512  032576  051122  051117  041520
5513  032604     000
5514  032605     105  050130  041505  DH23:   .ASCII  ?EXPECTD EXPECTD RECEIVD RECEIVD?<CRLF>
5515  032612  042124  042440  050130
5516  032620  041505  042124  051040
5517  032626  041505  044505  042126
5518  032634  051040  041505  044505
5519  032642  042126     200
5520  032645     040  047442  021122          .ASCIZ  ? "OR"    "AND"   "OR"    "AND"  #ERRORS TESTNO?
5521  032652  020040  020040  040442
```

```
5522  032660  042116  020042  020040
5523  032666  047442  021122  020040
5524  032674  020040  040442  042116
5525  032702  020042  021440  051105
5526  032710  047522  051522  052040
5527  032716  051505  047124  000117
5528  032724  042101  051104  051525  DH25:   .ASCIZ  ?ADDRUSED  BITDIFF TESTNO  ERRORPC?
5529  032732  042105  020040  044502
5530  032740  042124  043111  020106
5531  032746  042524  052123  047516
5532  032754  020040  051105  047522
5533  032762  050122  000103
5534  032766  042101  051104  051525  DH27:   .ASCIZ  ?ADDRUSED  TESTNO  ERRORPC?
5535  032774  042105  020040  042524
5536  033002  052123  047516  020040
5537  033010  051105  047522  050122
5538  033016  000103
5539  033020  042524  052123  047516  DH30:   .ASCIZ  ?TESTNO  ERRORPC?
5540  033026  020040  051105  047522
5541  033034  050122  000103
5542  033040  042522  047515  042526  DH34:   .ASCIZ  ?REMOVED MISSING TESTNO  ERRORPC?
5543  033046  020104  044515  051523
5544  033054  047111  020107  042524
5545  033062  052123  047516  020040
5546  033070  051105  047522  050122
5547  033076  000103
5548  033100  047514  042527  052123  DH35:   .ASCIZ  ?LOWEST  HIGHEST TESTNO  ERRORPC?
5549  033106  020040  044510  044107
5550  033114  051505  020124  042524
5551  033122  052123  047516  020040
5552  033130  051105  047522  050122
5553  033136  000103
5554  033140  042524  052123  047516  DH36:   .ASCIZ  ?TESTNO  ERRORPC UNIBUS ADDRESS OF MAP REGISTER UNDER TEST?
5555  033146  020040  051105  047522
5556  033154  050122  020103  047125
5557  033162  041111  051525  040440
5558  033170  042104  042522  051523
5559  033176  047440  020106  040515
5560  033204  020120  042522  044507
5561  033212  052123  051105  052440
5562  033220  042116  051105  052040
5563  033226  051505  000124
5564  033232  047503  042116  052111  DH40:   .ASCII  ?CONDITN CONDITN?<CRLF>
5565  033240  020116  047503  042116
5566  033246  052111  100116
5567  033252  054105  042520  052103          .ASCIZ  ?EXPECTD RECEIVD TESTNO  ERRORPC?
5568  033260  020104  042522  042503
5569  033266  053111  020104  042524
5570  033274  052123  047516  020040
5571  033302  051105  047522  050122
5572  033310  000103
5573  033312  047503  051122  041505  DH41:   .ASCII  ?CORRECT ADDRESS?<CRLF>
5574  033320  020124  042101  051104
5575  033326  051505  100123
5576  033332  042101  051104  051505          .ASCIZ  ?ADDRESS FETCHED TESTNO  ERRORPC?
5577  033340  020123  042506  041524
```

```
5578  033346  042510  020104  042524
5579  033354  052123  047516  020040
5580  033362  051105  047522  050122
5581  033370  000103
5582  033372  047503  051122  041505  DH42:   .ASCII  ?CORRECT EXPECTD RECEIVD?<CRLF>
5583  033400  020124  054105  042520
5584  033406  052103  020104  042522
5585  033414  042503  053111  100104
5586  033422  042101  051104  051505          .ASCIZ  ?ADDRESS DATA    FROM UB TESTNO  ERRORPC?
5587  033430  020123  040504  040524
5588  033436  020040  020040  051106
5589  033444  046517  052440  020102
5590  033452  042524  052123  047516
5591  033460  020040  051105  047522
5592  033466  050122  000103
5593  033472  044523  045132  046525  DH43:   .ASCIZ  ?SIZJUMP TOPFOUND TESTNO ERRORPC?
5594  033500  020120  047524  043120
5595  033506  052517  042116  052040
5596  033514  051505  047124  020117
5597  033522  051105  047522  050122
5598  033530  000103
5599  033532  047503  042116  052111  DH44:   .ASCII  ?CONDITN CONDITN ADDRESS   MAINTEN CONTROL?<CRLF>
5600  033540  020116  047503  042116
5601  033546  052111  020116  042101
5602  033554  051104  051505  020123
5603  033562  020040  040515  047111
5604  033570  042524  020116  047503
5605  033576  052116  047522  100114
5606  033604  054105  042520  052103          .ASCIZ  ?EXPECTD RECEIVD REFERENCD REGISTR REGISTR TESTNO  ERRORPC?
5607  033612  020104  042522  042503
5608  033620  053111  020104  042522
5609  033626  042506  042522  041516
5610  033634  020104  042522  044507
5611  033642  052123  020122  042522
5612  033650  044507  052123  020122
5613  033656  042524  052123  047516
5614  033664  020040  051105  047522
5615  033672  050122  000103
5616  033676  042522  040507  051104  DH201:  .ASCIZ  ?REGADRS TESTNO  ERRORPC?
5617  033704  020123  042524  052123
5618  033712  047516  020040  051105
5619  033720  047522  050122  000103
5620  033726  042522  040507  051104  DH202:  .ASCIZ  ?REGADRS DATAREC TESTNO  ERRORPC?
5621  033734  020123  040504  040524
5622  033742  042522  020103  042524
5623  033750  052123  047516  020040
5624  033756  051105  047522  050122
5625  033764  000103
5626  033766  040515  051120  043505  DH203:  .ASCII  ?MAPREG  MAPREG?<CRLF>
5627  033774  020040  040515  051120
5628  034002  043505     200
5629  034005     124  051505  044524          .ASCIZ  ?TESTING DUALED  TESTNO  ERRORPC?
5630  034012  043516  042040  040525
5631  034020  042514  020104  052040
5632  034026  051505  047124  020117
5633  034034  042440  051122  051117
```

```
5634  034042  041520     000
5635  034045     122  043505  042101  DH204:  .ASCIZ  ?REGADRS PATTERN EXPECTD RECEIVD TESTNO  ERRORPC?
5636  034052  051522  050040  052101
5637  034060  042524  047122  042440
5638  034066  050130  041505  042124
5639  034074  051040  041505  044505
5640  034102  042126  052040  051505
5641  034110  047124  020117  042440
5642  034116  051122  051117  041520
5643  034124     000
5644  034125     105  050130  041505  DH205:  .ASCIZ  ?EXPECTD RECEIVD ADDRSLOAD TESTNO  ERRORPC?
5645  034132  042124  051040  041505
5646  034140  044505  042126  040440
5647  034146  042104  051522  047514
5648  034154  042101  052040  051505
5649  034162  047124  020117  042440
5650  034170  051122  051117  041520
5651  034176     000
5652  034177     101  042104  042522  DH206:  .ASCII  ?ADDRESS ADDRESS?<CRLF>
5653  034204  051523  040440  042104
5654  034212  042522  051523     200
5655  034217     105  050130  041505          .ASCIZ  ?EXPECTD RECEIVD TESTNO  ERRORPC?
5656  034224  042124  051040  041505
5657  034232  044505  042126  052040
5658  034240  051505  047124  020117
5659  034246  042440  051122  051117
5660  034254  041520     000
5661  034257     122  043505  051511  DH207:  .ASCII  ?REGISTR EXPECTD RECEIVD?<CRLF>
5662  034264  051124  042440  050130
5663  034272  041505  042124  051040
5664  034300  041505  044505  042126
5665  034306     200
5666  034307     101  042104  042522          .ASCIZ  ?ADDRESS  DATA    DATA   TESTNO  ERRORPC?
5667  034314  051523  020040  040504
5668  034322  040524  020040  020040
5669  034330  040504  040524  020040
5670  034336  052040  051505  047124
5671  034344  020117  042440  051122
5672  034352  051117  041520     000
5673
5674
5675          034360                  .EVEN
5676  034360  001266  001264  001262  DT1:    .WORD   PCPUER,CPUEXP,TESTNO,BADPC,0
5677  034366  001302  000000
5678  034372  001266  001262  001302  DT2:    .WORD   PCPUER,TESTNO,BADPC,0
5679  034400  000000
5680  034402  001274  001270  001300  DT3:    .WORD   PPARER,PLOADR,PMAINT,PCONTR,TESTNO,BADPC,0
5681  034410  001276  001262  001302
5682  034416  000000
5683  034420  001154  001170  001262  DT4:    .WORD   $REGO,$TMPO,TESTNO,$ERRPC,0
5684  034426  001116  000000
5685  034432  001312  001314  001316  DT5:    .WORD   PMMR0,PMMR1,PMMR2,TESTNO,BADPC,0
5686  034440  001262  001302  000000
5687  034446  001226  001224  001254  DT6:    .WORD   ADDROR,ADRAND,ERRCNT,TESTNO,$ERRPC,0
5688  034454  001262  001116  000000
5689  034462  001226  001224  001232  DT10:   .WORD   ADDROR,ADRAND,DATAOR,DATAND,ERRCNT,TESTNO,$ERRPC,0
```

```
5690  034470  001230  001254  001262
5691  034476  001116  000000
5692  034502  001160  001154  001262  DT12:   .WORD   $REG2,$REG0,TESTNO,$ERRPC,0
5693  034510  001116  000000
5694  034514  001162  001156  001262  DT13:   .WORD   $REG3,$REG1,TESTNO,$ERRPC,0
5695  034522  001116  000000
5696  034526  001226  001224  001232  DT14:   .WORD   ADDROR,ADRAND,DATAOR,DATAND,ERRCNT,TESTNO,0
5697  034534  001230  001254  001262
5698  034542  000000
5699  034544  001226  001224  001236  DT15:   .WORD   ADDROR,ADRAND,PATTOR,PATAND,DATAOR,DATAND,ERRCNT,TESTNO,0
5700  034552  001234  001232  001230
5701  034560  001254  001262  000000
5702  034566  001154  001262  001116  DT17:   .WORD   $REG0,TESTNO,$ERRPC,0
5703  034574  000000
5704  034576  001156  001262  001116  DT20:   .WORD   $REG1,TESTNO,$ERRPC,0
5705  034604  000000
5706  034606  001236  001234  001232  DT23:   .WORD   PATTOR,PATAND,DATAOR,DATAND,ERRCNT,TESTNO,0
5707  034614  001230  001254  001262
5708  034622  000000
5709  034624  001164  001154  001262  DT25:   .WORD   $REG4,$REG0,TESTNO,$ERRPC,0
5710  034632  001116  000000
5711  034636  001156  001262  001116  DT27:   .WORD   $REG1,TESTNO,$ERRPC,0
5712  034644  000000
5713  034646  001262  001116  000000  DT30:   .WORD   TESTNO,$ERRPC,0
5714  034654  001254  001256  001262  DT34:   .WORD   ERRCNT,CNTR,TESTNO,$ERRPC,0
5715  034662  001116  000000
5716  034666  001240  001242  001262  DT35:   .WORD   LOWEST,HIGEST,TESTNO,$ERRPC,0
5717  034674  001116  000000
5718  034700  001262  001116  001154  DT36:   .WORD   TESTNO,$ERRPC,$REG0,0
5719  034706  000000
5720  034710  001226  001224  001232  DT37:   .WORD   ADDROR,ADRAND,DATAOR,DATAND,ERRCNT,TESTNO,0
5721  034716  001230  001254  001262
5722  034724  000000
5723  034726  001264  001266  001262  DT40:   .WORD   CPUEXP,PCPUER,TESTNO,$ERRPC,0
5724  034734  001116  000000
5725  034740  001160  001156  001262  DT41:   .WORD   $REG2,$REG1,TESTNO,$ERRPC,0
5726  034746  001116  000000
5727  034752  001156  001162  001160  DT42:   .WORD   $REG1,$REG3,$REG2,TESTNO,$ERRPC,0
5728  034760  001262  001116  000000
5729  034766  177760  001320  001262  DT43:   .WORD   SIZELO,RSIZE,TESTNO,$ERRPC,0
5730  034774  001116  000000
5731  035000  001200  001274  001270  DT44:   .WORD   $TMP4,PPARER,PLOADR,PMAINT,PCONTR,TESTNO,$ERRPC,0
5732  035006  001300  001276  001262
5733  035014  001116  000000
5734  035020  001154  001262  001116  DT201:  .WORD   $REG0,TESTNO,$ERRPC,0
5735  035026  000000
5736  035030  001154  001170  001262  DT202:  .WORD   $REG0,$TMP0,TESTNO,$ERRPC,0
5737  035036  001116  000000
5738  035042  001154  001156  001262  DT203:  .WORD   $REG0,$REG1,TESTNO,$ERRPC,0
5739  035050  001116  000000
5740  035054  001154  001160  001164  DT204:  .WORD   $REG0,$REG2,$REG4,$REG3,TESTNO,$ERRPC,0
5741  035062  001162  001262  001116
5742  035070  000000
5743  035072  001156  001154  001160  DT205:  .WORD   $REG1,$REG0,$REG2,TESTNO,$ERRPC,0
5744  035100  001262  001116  000000
5745  035106  001154  001162  001262  DT206:  .WORD   $REG0,$REG3,TESTNO,$ERRPC,0
```

```
5746 035114 001116 000000
5747 035120 001154 001160 001162 DT207: .WORD   $REG0,$REG2,$REG3,TESTNO,$ERRPC,0
5748 035126 001262 001116 000000
5749
5750
5751 035134    000    000    000 DF1:   .BYTE   0,0,0,0
5752 035137    000
5753 035140    000    000    000 DF2:   .BYTE   0,0,0
5754 035143    000    002    000 DF3:   .BYTE   0,2,0,0,0,0
5755 035146    000    000    000
5756 035151    000    000    000 DF5:   .BYTE   0,0,0,0,0
5757 035154    000    000
5758 035156    000    000    001 DF6:   .BYTE   0,0,1,0,0
5759 035161    000    000
5760 035163    000    000    000 DF10:  .BYTE   0,0,0,0,1,0,0
5761 035166    000    001    000
5762 035171    000
5763 035172    000    000    000 DF12:  .BYTE   0,0,0,0
5764 035175    000
5765 035176    000    000    000 DF14:  .BYTE   0,0,0,0,1,0
5766 035201    000    001    000
5767 035204    000    000    000 DF15:  .BYTE   0,0,0,0,0,0,1,0
5768 035207    000    000    000
5769 035212    001    000
5770 035214    000    000    000 DF17:  .BYTE   0,0,0
5771 035217    000    000    000 DF23:  .BYTE   0,0,0,0,1,0
5772 035222    000    001    000
5773 035225    003    004    000 DF25:  .BYTE   3,4,0,0
5774 035230    000
5775 035231    003    000    000 DF27:  .BYTE   3,0,0
5776 035234    000    000        DF30:  .BYTE   0,0
5777 035236    000    000    000 DF34:  .BYTE   0,0,0,0
5778 035241    000
5779 035242    004    004    000 DF35:  .BYTE   4,4,0,0
5780 035245    000
5781 035246    000    000    003 DF36:  .BYTE   0,0,3
5782 035251    000    000    000 DF37:  .BYTE   0,0,0,0,1,0
5783 035254    000    001    000
5784 035257    000    000    000 DF40:  .BYTE   0,0,0,0
5785 035262    000
5786 035263    000    000    000 DF41:  .BYTE   0,0,0,0
5787 035266    000
5788 035267    003    000    000 DF42:  .BYTE   3,0,0,0,0
5789 035272    000    000
5790 035274    000    000    000 DF43:  .BYTE   0,0,0,0
5791 035277    000
5792 035300    000    000    002 DF44:  .BYTE   0,0,2,0,0,0,0
5793 035303    000    000    000
5794 035306    000
5795 035307    000    000    000 DF201: .BYTE   0,0,0
5796 035312    000    000    000 DF202: .BYTE   0,0,0,0
5797 035315    000
5798 035316    000    000    000 DF203: .BYTE   0,0,0,0
5799 035321    000
5800 035322    000    000    000 DF204: .BYTE   0,0,0,0,0,0
5801 035325    000    000    000
```

```
 5802  035330      000      000      003  DF205:  .BYTE   0,0,3,0,0
 5803  035333      000      000
 5804  035335      000      000      000  DF206:  .BYTE   0,0,0,0
 5805  035340      000
 5806  035341      000      000      000  DF207:  .BYTE   0,0,0,0,0
 5807  035344      000      000
 5808                                       .EVEN
 5809
 5810              000001                   .END
```

```
ADDROR  001226          946#  1373*  2140*  2166*  2193*  2219*  3137*  3603*  5687  5689  5696  5699  5720
ADRAND  001224          945#  1377*  2142*  2168*  2195*  2221*  3141*  3598*  5687  5689  5696  5699  5720
BADPC   001302          974#  2324*  2370*  2416*  5676   5678   5680   5685
BIT0  = 000001          567#  3191
BIT00 = 000001          557#   567
BIT01 = 000002          556#   566
BIT02 = 000004          555#   565
BIT03 = 000010          554#   564
BIT04 = 000020          553#   563
BIT05 = 000040          552#   562
BIT06 = 000100          551#   561
BIT07 = 000200          550#   560
BIT08 = 000400          549#   559   1393
BIT09 = 001000          548#   558   1403   1476
BIT1  = 000002          566#  2379*
BIT10 = 002000          547#  1459
BIT11 = 004000          546#  1410   3216
BIT12 = 010000          545#  4813
BIT13 = 020000          544#  1466
BIT14 = 040000          543#
BIT15 = 100000          542#  4049   4120   4639   4712
BIT2  = 000004          565#  4038   4628
BIT3  = 000010          564#  4109   4701
BIT4  = 000020          563#  2071   3192   3210
BIT5  = 000040          562#  3255   3424   3560   3630   3850
BIT6  = 000100          561#
BIT7  = 000200          560#
BIT8  = 000400          559#
BIT9  = 001000          558#  1529
BPTVEC= 000014          574#
CACHVE= 000114          581#  2361*  2373*  2382*  2477*  2479*  3936*  3947*  3993*  4004*  4056*  4075*  4127*
                       4146*  4520*  4531*  4580*  4591*  4646*  4665*  4719*  4738*
CACOUN  005516         2277#  3126   3156
CLRMAP  005054         2108#  2832   3314   3509
CNTR    001256          964#  3452*  3456   3460*  5714
CONTRL= 177746          591#  2368   3059   3062*  3063   3115   3116   3118   3143*  3163*  3927   3928*  3948*
                       3956   3982   3983*  4005*  4013   4038   4040   4053*  4076*  4084   4109   4111   4124*
                       4147*  4155   4511   4512*  4532*  4540   4569   4570*  4592*  4600   4628   4630   4643*
                       4666*  4674   4701   4703   4716*  4739*  4747
COUNT   005352         2218#  2885   2929   2974   3018
CPFLAG  005574         1489*  2312#  2333*  2493*
CPUER   005572         2311#  2483   2591
CPUERR= 177766          604#  1485*  2134   2135*  2323   2332*  2498*
CPUEXP  001264          967#  2136   2325   2327   2489*  3211*  3234*  3251*  3289*  3315*  3353*  3374*  3392*
                       3418*  3461*  3631*  3649*  3860*  3871*  3901*  4210*  4228*  4444*  4455*  4485*  5676
                       5723
CR    = 000015          479#  1778   1788
CRLF  = 000200          480#  1752   1788   2476   4972   4987   5003   5021   5072   5086   5121   5141   5159
                       5168   5177   5259   5275   5290   5306   5321   5412   5428   5441   5451   5467   5476
                       5490   5514   5564   5573   5582   5599   5626   5652   5661
DATA    001326          985#  3862   3886*  3893*  3894*  3895*  4446   4470*  4477*  4478*  4479*
DATAND  001230          947#  1376*  2172*  2199*  2225*  2256*  2285*  3140*  3601*  5689   5696   5699   5706
                       5720
DATAOR  001232          948#  1372*  2170*  2197*  2223*  2254*  2283*  3136*  3604*  5689   5696   5699   5706
                       5720
DF1     035134         1007   5751#
```

DEKBF8.P11 CROSS REFERENCE TABLE -- USER SYMBOLS

```
DF10    035163          1057    1064    5760#
DF12    035172          1071    1078    5763#
DF14    035176          1085    5765#
DF15    035204          1092    1099    5767#
DF17    035214          1106    1113    1120    1127    5770#
DF2     035140          1013    5753#
DF201   035307          1310    5795#
DF202   035312          1316    5796#
DF203   035316          1323    5798#
DF204   035322          1329    5800#
DF205   035330          1335    5802#
DF206   035335          1342    5804#
DF207   035341          1349    5806#
DF23    035217          1134    1141    1176    5771#
DF25    035225          1148    1155    5773#
DF27    035231          1162    5775#
DF3     035143          1021    1029    5754#
DF30    035234          1169    1182    1190    5776#
DF34    035236          1197    5777#
DF35    035242          1206    5779#
DF36    035246          1212    5781#
DF37    035251          1219    1300    5782#
DF40    035257          1226    1261    5784#
DF41    035263          1233    1269    5786#
DF42    035267          1240    1277    5788#
DF43    035274          1246    1284    5790#
DF44    035300          1253    1293    5792#
DF5     035151          1036    5756#
DF6     035156          1043    1050    5758#
DH1     031425          1005    5400#
DH10    032051          1054    1061    5451#
DH12    032201          1068    1075    5467#
DH14    032257          1082    5476#
DH15    032376          1089    1096    5490#
DH17    032555          1104    1111    1118    1125    5509#
DH2     031471          1011    5407#
DH201   033676          1308    5616#
DH202   033726          1314    5620#
DH203   033766          1320    5626#
DH204   034045          1327    5635#
DH205   034125          1333    5644#
DH206   034177          1339    5652#
DH207   034257          1346    5661#
DH23    032605          1131    1138    1173    1216    1297    5514#
DH25    032724          1146    1153    5528#
DH27    032766          1160    5534#
DH3     031525          1018    1026    5412#
DH30    033020          1167    1180.   1188    5539#
DH34    033040          1195    5542#
DH35    033100          1204    5548#
DH36    033140          1210    5554#
DH40    033232          1223    1258    5564#
DH41    033312          1230    1266    5573#
DH42    033372          1237    1274    5582#
DH43    033472          1244    1282    5593#
DH44    033532          1250    1290    5599#
```

```
DH5     031655          1033    5428#
DH6     031761          1040    1047    5441#
DISPLA= 177570           474#   1424*   1455*   3209*   3417*   4208*
DT1     034360          1006    5676#
DT10    034462          1056    1063    5689#
DT12    034502          1070    5692#
DT13    034514          1077    5694#
DT14    034526          1084    5696#
DT15    034544          1091    1098    5699#
DT17    034566          1105    1119    1126    5702#
DT2     034372          1012    5678#
DT20    034576          1112    5704#
DT201   035020          1309    5734#
DT202   035030          1315    5736#
DT203   035042          1322    5738#
DT204   035054          1328    5740#
DT205   035072          1334    5743#
DT206   035106          1341    5745#
DT207   035120          1348    5747#
DT23    034606          1133    1140    1175    5706#
DT25    034624          1147    1154    5709#
DT27    034636          1161    5711#
DT3     034402          1020    1028    5680#
DT30    034646          1168    1181    1189    5713#
DT34    034654          1196    5714#
DT35    034666          1205    5716#
DT36    034700          1211    5718#
DT37    034710          1218    1299    5720#
DT4     034420          5683#
DT40    034726          1225    1260    5723#
DT41    034740          1232    1268    5725#
DT42    034752          1239    1276    5727#
DT43    034766          1245    1283    5729#
DT44    035000          1252    1292    5731#
DT5     034432          1035    5685#
DT6     034446          1042    1049    5687#
DUALAD  005276          2191#   2840    3525
EMTVEC= 000030           577#   2445*   2446*
EM1     023162          1004    4830#
EM10    023773          1053    4899#
EM11    024064          1060    4909#
EM12    024156          1067    4919#
EM13    024244          1074    4928#
EM14    024336          1081    4938#
EM15    024431          1088    4948#
EM16    024535          1095    4960#
EM17    024640          1102    4972#
EM2     023244          1010    4839#
EM20    024762          1109    4987#
EM201   030720          1307    5341#
EM202   030774          1313    5349#
EM203   031047          1319    5357#
EM204   031142          1326    5367#
EM205   031222          1332    5375#
EM206   031271          1338    5382#
EM207   031353          1345    5391#
```

DEKBFB.P11 CROSS REFERENCE TABLE -- USER SYMBOLS

```
EM21    025110            1116    5003#
EM22    025254            1123    5021#
EM23    025420            1130    5039#
EM24    025514            1137    5049#
EM25    025614            1144    5060#
EM26    025721            1151    1158    5072#
EM3     023313            1016    4846#
EM30    026042            1165    5086#
EM31    026216            1172    5104#
EM32    026310            1179    5114#
EM33    026362            1185    5121#
EM34    026537            1193    5141#
EM35    026710            1200    5159#
EM36    027175            1209    5190#
EM37    027272            1215    5201#
EM4     023414            1024    4857#
EM40    027375            1222    5213#
EM41    027470            1229    5223#
EM42    027552            1236    5232#
EM43    027645            1243    5242#
EM44    027732            1249    5251#
EM45    030007            1256    5259#
EM46    030142            1264    5275#
EM47    030264            1272    5290#
EM5     023523            1032    4869#
EM50    030417            1280    5306#
EM51    030544            1288    5321#
EM52    030661            1296    5335#
EM6     023631            1039    4881#
EM7     023711            1046    4890#
EREXIT  002760            1484    1492#
ERRCNT  001254             963#   1371*   1446*   2144    2174    2201    2231    2258    2531    2562    2604    2638    2672
                          2706    2849    2896    2940    2985    3029    3130    3135*   3159    3348    3448*   3456    3459*
                          3540    3605    3611    5687    5689    5696    5699    5706    5714    5720
ERRVEC= 000004             570#   1384    1385*   1387*   1390*   2483*   2484*   2521*   2591*
ERTYPE  002762            1468    1519#
ER200   002050            1302#   1539    1542
FLAG    001260             965#
GNS     = ****** U         857    1959    1960    1961    1962    1963    1964    1965    1966    1967    2475    4782    4789
HIADRS= 177742             589#   2366    3082    3954    4011    4082    4153    4538    4598    4672    4745
HIGEST  001242             953#   3439*   3466    3473    3529    3538    5716
HITMIS= 177752             593#   2558
HREGL   001250             959#   3484*
HREGU   001252             961#   3486*
HT      = 000011           477#   1750    1788
IOTVEC= 000020             575#   2443*   2444*
KDPAR0= 172360             747#
KDPAR1= 172362             748#
KDPAR2= 172364             749#
KDPAR3= 172366             750#
KDPAR4= 172370             751#
KDPAR5= 172372             752#
KDPAR6= 172374             753#
KDPAR7= 172376             754#
KDPDR0= 172320             725#
KDPDR1= 172322             726#
```

```
KDPDR2= 172324          727#
KDPDR3= 172326          728#
KDPDR4= 172330          729#
KDPDR5= 172332          730#
KDPDR6= 172334          731#
KDPDR7= 172336          732#
KERSTK= 001100          464#  2321
KIPAR0= 172340          736#  1631  3183*  4189*
KIPAR1= 172342          737#  3184* 4187*
KIPAR2= 172344          738#  3185*
KIPAR3= 172346          739#  3186*
KIPAR4= 172350          740#  3187* 3426* 3433* 3434  3438  3439  3440* 3441  3453* 3454  3519* 3561*
                       3568* 3569  3632* 3857* 3885  3887* 4211* 4441* 4469  4471*
KIPAR5= 172352          741#  3188* 3328* 3329*
KIPAR6= 172354          742#  3189* 3213* 3218* 3229* 3253* 3257* 3264* 3317* 3323* 3324  3328  3376
                       3576  3579  3591* 3592  3674* 3689* 3704* 3719* 3734* 3749* 3764* 3779* 3794*
                       3809* 3824* 3858* 3929* 3984* 4044* 4115* 4258* 4273* 4288* 4303* 4318* 4333*
                       4348* 4363* 4378* 4393* 4408* 4442* 4513* 4571* 4634* 4707*
KIPAR7= 172356          743#  3190*
KIPDR0= 172300          714#  3175*
KIPDR1= 172302          715#  3176*
KIPDR2= 172304          716#  3177*
KIPDR3= 172306          717#  3178*
KIPDR4= 172310          718#  3179*
KIPDR5= 172312          719#  3180*
KIPDR6= 172314          720#  3181*
KIPDR7= 172316          721#  3182*
KSP    =X000006         498#  1519* 1523* 1535* 1563* 1570* 1576* 1582* 1584* 1585  1593* 1598* 1606
                       1607  1624  1639* 1641* 1642  1647* 1655  1660* 1662* 1663  1666* 2073  2075
                       2076* 2092* 2132  2133  2150* 2151* 2165  2192  2218  2249  2278  2319  2320
                       2321* 2334* 2335* 2363  2364  2376* 2381  2393* 2394* 2416  2417  2418  2425*
                       2426*
LF     = 000012         478#  1782  1788
LOADRS= 177740          588#  2365  2553  3075  3953  4010  4081  4152  4537  4597  4671  4744
LOOP    010346         2477#  4823
LOWEST  001240          951#  3438* 3464  3474  3514  3515  3632  3674  3689  3704  3719  3734  3749
                       3764  3779  3794  3809  3824  3858  3929  3984  4044  4115  4185  4211  4259
                       4273  4288  4303  4318  4333  4348  4363  4378  4393  4408  4442  4513  4571
                       4634  4707  5716
LREGL   001244          955#  3481* 3513  3634* 3673* 3688* 3703* 3718* 3733* 3748* 3763* 3778* 3793*
                       3808* 3823* 3854* 3888* 3930* 3985* 4046* 4117* 4213* 4257* 4272* 4287* 4302*
                       4317* 4332* 4347* 4362* 4377* 4392* 4407* 4438* 4472* 4514* 4572* 4636* 4709*
LREGU   001246          957#  3483* 3633* 3672* 3687* 3702* 3717* 3732* 3747* 3762* 3777* 3792* 3807*
                       3822* 3853* 3890* 3891  3931* 3986* 4045* 4116* 4179  4212* 4256* 4271* 4286*
                       4301* 4316* 4331* 4346* 4361* 4376* 4391* 4406* 4437* 4474* 4475  4515* 4573*
                       4635* 4708*
MAINT = 177750          592#  2369  3068* 3069  3144  3932  3937* 3987  3994* 4047  4057* 4118  4129*
                       4516  4521* 4574  4581* 4637  4647* 4710  4720*
MAPHO = 170202          831#  2626  2785*
MAPH00= 170202          767#   831  2786  2923
MAPH01= 170206          769#   833
MAPH02= 170212          771#   835
MAPH03= 170216          773#   837
MAPH04= 170222          775#   839
MAPH05= 170226          777#   841
MAPH06= 170232          779#   843
```

```
MAPH07= 170236           781#   845
MAPH1 = 170206           833#
MAPH10= 170242           783#
MAPH11= 170246           785#
MAPH12= 170252           787#
MAPH13= 170256           789#
MAPH14= 170262           791#
MAPH15= 170266           793#
MAPH16= 170272           795#
MAPH17= 170276           797#   2634    2932
MAPH2 = 170212           835#
MAPH20= 170302           799#   2694    2797*   2798    3012
MAPH21= 170306           801#
MAPH22= 170312           903#
MAPH23= 170316           905#
MAPH24= 170320           807#
MAPH25= 170326           809#
MAPH26= 170332           811#
MAPH27= 170336           813#
MAPH3 = 170216           837#
MAPH30= 170342           815#
MAPH31= 170346           817#
MAPH32= 170352           819#
MAPH33= 170356           821#
MAPH34= 170362           823#
MAPH35= 170366           825#
MAPH36= 170372           827#
MAPH37= 170376           829#   2110    2527    2702    2831    2845    3021
MAPH4 = 170222           839#
MAPH5 = 170226           841#
MAPH6 = 170232           843#
MAPH7 = 170236           845#
MAPL0 = 170200           830#   2108    2522    2592    2829    2842    3215*   3224    3419
MAPL00= 170200           766#    830    2742*   2743    2879
MAPL01= 170204           768#    832
MAPL02= 170210           770#    834
MAPL03= 170214           772#    836
MAPL04= 170220           774#    838
MAPL05= 170224           776#    840
MAPL06= 170230           778#    842
MAPL07= 170234           780#    844
MAPL1 = 170204           832#
MAPL10= 170240           782#
MAPL11= 170244           784#
MAPL12= 170250           786#
MAPL13= 170254           788#
MAPL14= 170260           790#
MAPL15= 170264           792#
MAPL16= 170270           794#
MAPL17= 170274           796#   2600    2888
MAPL2 = 170210           834#
MAPL20= 170300           798#   2660    2754*   2755    2968
MAPL21= 170304           800#
MAPL22= 170310           802#
MAPL23= 170314           804#
MAPL24= 170320           806#
```

```
MAPL25= 170324            808*
MAPL26= 170330            810*
MAPL27= 170334            812*
MAPL3 = 170214            836*
MAPL30= 170340            814*
MAPL31= 170344            816*
MAPL32= 170350            818*
MAPL33= 170354            820*
MAPL34= 170360            822*
MAPL35= 170364            824*
MAPL36= 170370            826*
MAPL37= 170374            828*   2668    2977    3422
MAPL4 = 170220            838*
MAPL5 = 170224            840*
MAPL6 = 170230            842*
MAPL7 = 170234            844*
MEMER   005704           2353*   2373    2382    2477    3947    4004    4075    4146    4531    4591    4665    4738
MEMERR= 177744            590*   1486*   2367    2391*   2497*   3073*   3080*   3926*   3946*   3955    3981*   4003*   4012
                         4043*   4074*   4083    4114*   4145*   4154    4510*   4530*   4539    4568*   4590*   4599    4633*
                         4664*   4673    4706*   4737*   4746
MMFLAG  006132           1491*   2409*   2424*   2496*
MMR0  = 177572            615*    619    1487*   2419    2423*   2490*   3191*   4767*
MMR1  = 177574            616*    620    2420
MMR2  = 177576            617*    621    2421
MMR3  = 172516            618*    622    2491*   3192*   3424*   3560*   3630*   4768*
MMTRAP  006130           2408*   2480
MMVEC = 000250            583*   2480*   2482*
NEXMEM= 000040           3850*   3871    3873    4455    4457
NXTTST  001324            983*   2389    2519*   2551*   2589*   2624*   2658*   2692*   2735*   2779*   2826*   2872*   2916*
                         2961*   3005*   3055*   3111*   3249*   3311*   3372*   3410*   3506*   3558*   3628*   3665*   3847*
                         3921*   3977*   4035*   4106*   4209*   4247*   4433*   4505*   4564*   4625*   4698*
OLDPC   001304            975*   2132*   2151    2165*   2179    2192*   2206    2218*   2236    2249*   2263    2278*   2292
                         2319*   2324    2335    2363*   2370    2387*   2389*   2394    2417*   2426
OLDPS   001306            976*   2133*   2150    2320*   2334    2364*   2393    2418*   2425
OLDPSW  001310            977*   2075*   2092    2093*
PADRSH  001222            943*   1637*   1659*
PADRSL  001220            941*   1638*   1639    1658*   1660
PAFLAG  005706           1490*   2354*   2392*   2495*
PATAND  001234            949*   1378*   2229*   2252*   2281*   3142*   5699    5706
PATTOR  001236            950*   1374*   2227*   2250*   2279*   3138*   5699    5706
PC     =%000007           501*   1468*   1583*   1594*   1599*   1608*   1640*   1648*   1661*   1667*   1757*   1764*   1770*
                         1784*   1786*   2002    2044*   2112*   2124*   2311*   2353*   2408*   2598*   2632*   2666*   2700*
                         2832*   2840*   2885*   2929*   2974*   3018*   3126*   3156*   3314*   3345*   3509*   3525*   3882*
                         4067*   4138*   4466*   4657*   4730*   4774*   4777*   4806*
PCONTR  001276            972*   2368*   3956*   4013*   4084*   4155*   4540*   4600*   4674*   4747*   5680    5731
PCPUER  001266            968*   2134*   2135    2136    2323*   2327    2332    3318*   3321    3636*   3645    3861*   3869
                         3873    4215*   4224    4445*   4453    4457    5676    5678    5723
PHIADR  001272            970*   2366*   3954*   4011*   4082*   4153*   4538*   4598*   4672*   4745*
PIRQ  = 177772            472*
PIRQVE= 000240            582*
PLOADR  001270            969*   2365*   2377    3953*   4010*   4081*   4152*   4537*   4597*   4671*   4744*   5680    5731
PMAINT  001300            973*   2369*   3957*   4014*   4085*   4156*   4541*   4601*   4675*   4748*   5680    5731
PMMR0   001312            978*   2419*   5685
PMMR1   001314            979*   2420*   5685
PMMR2   001316            980*   2421*   5685
PPARER  001274            971*   2367*   2371    2391    3925*   3943    3955*   3980*   4000    4012*   4042*   4070    4083*
```

```
                        4113*   4141    4154*   4509*   4527    4539*   4567*   4587    4599*   4632*   4660    4673*   4705*
                        4733    4746*   5680    5731
PR0     = 000000         504#
PR1     = 000040         505#
PR2     = 000100         506#
PR3     = 000140         507#
PR4     = 000200         508#
PR5     = 000240         509#
PR6     = 000300         510#
PR7     = 000340         511#
PS      = 177776         469#    470     2437*   4811
PSW     = 177776         470#
PWRMSG  004624          2001    2008#
PWRVEC= 000024           576#   1974*   1975*   1983*   1998*   1999*   2449*   2450*
RESREG= 104414          1646    1965#   2043
RESVEC= 000010           571#   2458*   2465*
RETRY   001322           982#   1369*   2384    2386*   3133*
RSIZE   001320           981#   3885*   3897    4469*   4481    5729
R0      =%000000         483#    491    1375*   1376    1377    1378    1447    1519    1520*   1521*   1526*   1527*   1529
                        1538*   1539*   1541*   1542*   1546*   1547    1554    1564    1566*   1567    1573    1579    1589
                        1601*   1607*   1625*   1626*   1627*   1631*   1632    1656*   1657*   1659    1689    1714*   1742
                        1743*   1744    1747*   1880    1890*   1894    1910    1911    1924*   1943    1944*   1945    1946*
                        1947*   1948*   1976    1997*   2033*   2037    2047*   2049*   2051*   2140    2141*   2142    2143*
                        2166    2167*   2168    2169*   2193    2194*   2195    2196*   2219    2220*   2221    2222*   2254
                        2255*   2256    2257*   2376    2377*   2378*   2379    2380    2381*   2492*   2493    2494    2495
                        2496    2497    2498    2522*   2525    2526*   2527    2553*   2556    2557*   2558    2592*   2595*
                        2596    2599*   2600    2626*   2629*   2630    2633*   2634    2660*   2663*   2664    2667*   2668
                        2694*   2697*   2698    2701*   2702    2743*   2744    2786*   2787    2829*   2835*   2838    2844*
                        2845    2879*   2881*   2882    2887*   2888    2923*   2925*   2926    2931*   2932    2968*   2970*
                        2971    2976*   2977    3012*   3014*   3015    3020*   3021    3058*   3063*   3075*   3076    3082*
                        3083*   3084    3118*   3122*   3123    3139*   3140    3141    3142    3144*   3145    3150*   3151
                        3153*   3173*   3175    3176    3177    3178    3179    3180    3181    3182    3216*   3218    3229
                        3230*   3231    3255*   3257    3264    3265*   3320*   3334    3342*   3343    3376*   3377*   3378*
                        3379*   3380*   3419*   3420*   3421*   3422    3425*   3430*   3444*   3449*   3473*   3475*   3477*
                        3480*   3484    3485*   3486    3514*   3523    3537*   3538    3562*   3565    3566*   3579*   3581*
                        3586    3589*   3597*   3598    3599*   3603    3675*   3677    3690*   3692    3705*   3707    3720*
                        3722    3735*   3737    3750*   3752    3765*   3767    3780*   3782    3795*   3797    3810*   3812
                        3825*   3827    3856*   3862*   3876    3934*   3935*   3939    3942*   3990*   3992*   3996    3999*
                        4259*   4261    4274*   4276    4289*   4291    4304*   4306    4319*   4321    4334*   4336    4349*
                        4351    4364*   4366    4379*   4381    4394*   4396    4409*   4411    4440*   4446*   4460    4518*
                        4519*   4523    4526*   4577*   4579*   4583    4586*   4796*   4803*   4806
R1      =%000001         484#    492    1448    1563    1564*   1570    1576    1582    1593    1598    1603    1606*   1624*
                        1628*   1629*   1630*   1635    1655*   1658    1690    1713*   1881    1894*   1895    1899    1923*
                        1977    1996*   2029*   2033    2034*   2046*   2048*   2050*   2197    2198*   2199    2200*   2250
                        2251*   2252    2253*   2755*   2756    2798*   2799    2831*   2836    2838    2841*   2842    3069*
                        3145*   3152*   3220*   3221    3228    3259*   3261    3268*   3270    3275*   3277    3282*   3284
                        3331*   3332    3334    3337*   3341    3343    3346*   3427*   3433    3440    3453    3474*   3476*
                        3478*   3479*   3481    3482*   3483    3515*   3519    3523    3528*   3529    3563*   3565*   3582*
                        3585    3588*   3677*   3679    3692*   3694    3707*   3709    3722*   3724    3737*   3739    3752*
                        3754    3767*   3769    3782*   3784    3797*   3799    3812*   3814    3827*   3829    3855*   3872
                        3932*   3940    3941*   3987*   3997    3998*   4054*   4055*   4057    4060*   4063*   4064*   4125*
                        4126*   4128    4131*   4134*   4135*   4179*   4180*   4181*   4182*   4184*   4185*   4186*   4187
                        4188*   4189    4261*   4263    4276*   4278    4291*   4293    4306*   4308    4321*   4323    4336*
                        4338    4351*   4353    4366*   4368    4381*   4383    4396*   4398    4411*   4413    4439*   4456
                        4516*   4524    4525*   4574*   4584    4585*   4644*   4645*   4647    4650*   4653*   4654*   4717*
                        4718*   4720    4723*   4726*   4727*
```

```
R10     =%000000          491#
R11     =%000001          492#
R12     =%000002          493#
R13     =%000003          494#
R14     =%000004          495#
R15     =%000005          496#
R2      =%000002          485#   493   1449  1633* 1634* 1636* 1637  1691  1712* 1882  1893* 1897* 1900
                         1907*  1908*  1909  1914* 1922* 1978  1995* 2035* 2036  2037* 2052* 2053* 2279
                         2280*  2281   2282* 2525* 2556* 2739* 2742  2744  2749* 2781* 2785  2787  2791*
                         2792   2836*  2875* 2877  2881  2890  2892* 2919* 2921  2925  2934  2936* 2964*
                         2966   2970   2979  2981* 3008* 3010  3014  3023  3025* 3119* 3122  3124  3127*
                         3128   3146*  3148  3154  3157* 3214* 3215  3221  3254* 3261  3271  3278  3285
                         3339*  3341*  3342  3428* 3430  3431  3444  3445  3449  3450  3513* 3516* 3517
                         3521   3535*  3536* 3564* 3567* 3583* 3590* 3678* 3679  3693* 3694  3708* 3709
                         3723*  3724   3738* 3739  3753* 3754  3768* 3769  3783* 3784  3798* 3799  3813*
                         3814   3828*  3829  3876* 3877  3940* 3997* 4072* 4143* 4262* 4263  4277* 4278
                         4292*  4293   4307* 4308  4322* 4323  4337* 4338  4352* 4353  4367* 4368  4382*
                         4383   4397*  4398  4412* 4413  4460* 4461  4524* 4584* 4662* 4735*
R3      =%000003          486#   494   1450  1632* 1635* 1638  1692  1711* 1822  1831* 1837* 1838* 1841*
                         1846*  1847*  1848  1857* 1883  1891* 1892* 1906* 1909* 1918* 1919* 1921* 1979
                         1994*  2032*  2045* 2052  2108* 2109* 2110  2223  2224* 2225  2226* 2283  2284*
                         2285   2286*  2738* 2754  2756  2758* 2782* 2797  2799  2801* 2802  2882* 2883
                         2886*  2926*  2927  2930* 2971* 2972  2975* 3015* 3016  3019* 3123* 3124  3151*
                         3154   3270*  3271  3277* 3278  3284* 3285  3387* 3388  3511* 3520  3585* 3586
                         3600*  3601   3602* 3604  3638* 3872* 3877  3939* 3996* 4061* 4072  4132* 4143
                         4217*  4456*  4461  4523* 4583* 4651* 4662  4724* 4735
R4      =%000004          487#   495   1451  1693  1710* 1823  1825* 1826* 1827* 1828  1829* 1843  1845*
                         1853*  1856*  1980  1993* 2031* 2038* 2227  2228* 2229  2230* 2877* 2878* 2883
                         2921*  2922*  2927  2966* 2967* 2972  3010* 3011* 3016  3217* 3220  3223* 3228*
                         3256*  3259   3520* 3521  4048* 4049* 4054  4060  4119* 4120* 4125  4131  4638*
                         4639*  4644   4650  4711* 4712* 4717  4723
R5      =%000005          488#   496   1452  1694  1709* 1824  1830* 1832* 1834* 1835* 1836* 1837  1855*
                         1884   1886*  1888* 1895* 1899* 1914  1920* 1981  1992* 2030* 2036* 2041* 2042
                         3148*  3150   4047* 4061  4062* 4118* 4132  4133* 4637* 4651  4652* 4710* 4724
                         4725*
R6      =%000006          489#   497   2438* 2439* 2440
R7      =%000007          490#   501
SAVREG= 104412           1623   1964#  2028
SDPAR0= 172260            703#
SDPAR1= 172262            704#
SDPAR2= 172264            705#
SDPAR3= 172266            706#
SDPAR4= 172270            707#
SDPAR5= 172272            708#
SDPAR6= 172274            709#
SDPAR7= 172276            710#
SDPDR0= 172220            681#
SDPDR1= 172222            682#
SDPDR2= 172224            683#
SDPDR3= 172226            684#
SDPDR4= 172230            685#
SDPDR5= 172232            686#
SDPDR6= 172234            687#
SDPDR7= 172236            688#
SIPAR0= 172240            692#
SIPAR1= 172242            693#
```

```
SIPAR2= 172244            694#
SIPAR3= 172246            695#
SIPAR4= 172250            696#
SIPAR5= 172252            697#
SIPAR6= 172254            698#
SIPAR7= 172256            699#
SIPDR0= 172200            670#
SIPDR1= 172202            671#
SIPDR2= 172204            672#
SIPDR3= 172206            673#
SIPDR4= 172210            674#
SIPDR5= 172212            675#
SIPDR6= 172214            676#
SIPDR7= 172216            677#
SIZEHI= 177762            601#
SIZEJ   014062           3327    3413#
SIZEL0= 177760            599#    3897    4481    5729
SP     =%000006           497#     498     499     500  1384*  1387  1389  1390  1395*  1396*  1397  1420  1421
                         1425*   1463   1478*  1481*  1689*  1690*  1691*  1692*  1693*  1694*  1695*  1696*  1697*
                         1698*   1705*  1706*  1707*  1708*  1709   1710   1711   1712   1713   1714   1742*  1743
                         1744*   1746   1747   1748*  1750   1752   1754   1758   1760*  1762*  1769*  1773   1777
                         1778    1782   1814*  1815   1816   1817*  1822*  1823*  1824*  1830   1855   1856   1857
                         1858*   1859*  1880*  1881*  1882*  1883*  1884*  1885*  1886   1889*  1902   1904*  1906
                         1916    1918   1920   1921   1922   1923   1924   1926*  1927*  1943*  1944   1976*  1977*
                         1978*   1979*  1980*  1981*  1982   1988*  1992   1993   1994   1995   1996   1997   2002*
                         2029    2042*  2442*  2459*  2460*  2464*  4784*  4791*  4798*  4799*  4811*  4812*  4817*
                         4818*
SR0    = 177572           619#
SR1    = 177574           620#
SR2    = 177576           621#
SR3    = 172516           622#
SSP    =%000006           499#                                                                          —
STACK  = 001100           463#     464     465     466  2442
START    010000           861     2003    2436#   3437
STKLMT= 177774            471#
SUPSTK= 000700            465#
SWR    = 177570           473#     474    1379*   1393   1395   1403   1410   1456   1459   1466   1470   1476   1483
                         1532    4813
SW0    = 000001           539#
SW00   = 000001           529#     539
SW01   = 000002           528#     538
SW02   = 000004           527#     537
SW03   = 000010           526#     536
SW04   = 000020           525#     535
SW05   = 000040           524#     534
SW06   = 000100           523#     533
SW07   = 000200           522#     532
SW08   = 000400           521#     531
SW09   = 001000           520#     530
SW1    = 000002           538#
SW10   = 002000           519#
SW11   = 004000           518#
SW12   = 010000           517#
SW13   = 020000           516#
SW14   = 040000           515#
SW15   = 100000           514#
```

```
SW2    = 000004            537#
SW3    = 000010            536#
SW4    = 000020            535#
SW5    = 000040            534#
SW6    = 000100            533#
SW7    = 000200            532#   1532
SW8    = 000400            531#
SW9    = 001000            530#   1483
SYSTID= 177764             603#
TBIT   = 000020           2071#   2073   2076   2093
TBITO  = 104416           1966#   3114   3923   4507
TBITOF  005010            1966    2072#
TBITR  = 104420           1967#   3171   4167   4769
TBITRE  005036            1967    2091#
TBITVE= 000014             572#   2455*  2456*
TESTNO  001262             966#   1445*  5676   5678   5680   5683   5685   5687   5689   5692   5694   5696   5699
                          5702    5704   5706   5709   5711   5713   5714   5716   5718   5720   5723   5725   5727
                          5729    5731   5734   5736   5738   5740   5743   5745   5747
TIMEOU  005072            2123#   2521
TIMOUT= 000020            3210#   3211   3251   3315   3374   3418   3631   3645   4444
TKVEC  = 000060            579#
TOFLAG  005074            1488*   2125#  2149*  2494*
TPVEC  = 000064            580#
TRAPVE= 000034             578#   2447*  2448*
TRTVEC= 000014             573#
TST1    010470            2517#
TST10   011324            2735    2777#
TST11   011446            2779    2824#
TST12   011572            2826    2850   2870#
TST13   011716            2872    2897   2914#
TST14   012042            2916    2941   2959#
TST15   012166            2961    2986   3003#
TST16   012312            3005    3030   3053#
TST17   012502            3055    3109#
TST2    010554            2519    2532   2549#
TST20   013124            3111    3205#
TST21   013276            3247#
TST22   013500            3249    3309#
TST23   013732            3311    3370#
TST24   014044            3372    3408#
TST25   014500            3410    3463   3504#
TST26   014674            3506    3541   3556#
TST27   015154            3558    3578   3626#
TST3    010632            2551    2563   2587#
TST30   015266            3628    3663#
TST31   016174            3665    3845#
TST32   016464            3847    3919#
TST33   016726            3921    3950   3975#
TST34   017200            3977    4007   4033#
TST35   017466            4035    4039   4078   4104#
TST36   020032            4106    4204#
TST37   020164            4209    4245#
TST4    010730            2589    2605   2622#
TST40   021100            4247    4431#
TST41   021370            4433    4503#
TST42   021632            4505    4534   4562#
```

DEKBFB.P11 CROSS REFERENCE TABLE -- USER SYMBOLS

```
TST43   022104            4564    4594    4623#
TST44   022372            4625    4629    4668    4697#
TST5    011020            2624    2639    2656#
TST6    011110            2658    2673    2690#
TST7    011200            2692    2707    2733#
TYPDS = 104410            1577    1963#   4786    4793
TYPE  = 104400            1461    1469    1549    1551    1556    1558    1586    1602    1643    1664    1755    1849    1925
                          1959#   2000    2473    4780    4787    4794
TYPOC = 104402            1524    1571    1960#
TYPON = 104406            1962#
TYPOS = 104404            1961#
TYPVAD  003270            1594    1623#
UBADDR  003376            1599    1655#
UBCOUN  005442            2249#   3345
UBMAP   017754            4110    4149    4166#
UDPAR0= 177660             659#
UDPAR1= 177662             660#
UDPAR2= 177664             661#
UDPAR3= 177666             662#
UDPAR4= 177670             663#
UDPAR5= 177672             664#
UDPAR6= 177674             665#
UDPAR7= 177676             666#
UDPDR0= 177620             637#
UDPDR1= 177622             638#
UDPDR2= 177624             639#
UDPDR3= 177626             640#
UDPDR4= 177630             641#
UDPDR5= 177632             642#
UDPDR6= 177634             643#
UDPDR7= 177636             644#
UIPAR0= 177640             648#
UIPAR1= 177642             649#
UIPAR2= 177644             650#
UIPAR3= 177646             651#
UIPAR4= 177650             652#
UIPAR5= 177652             653#
UIPAR6= 177654             654#
UIPAR7= 177656             655#
UIPDR0= 177600             626#
UIPDR1= 177602             627#
UIPDR2= 177604             628#
UIPDR3= 177606             629#
UIPDR4= 177610             630#
UIPDR5= 177612             631#
UIPDR6= 177614             632#
UIPDR7= 177616             633#
USESTK= 000600             466#
USP    =%000006            500#
WRITEE  005212            2164#   2598    2632    2666    2700
$BDADR  001122             909#
$BDDAT  001126             911#
$BELL   001210             937#   1461    1493
$CHARC  003750            1771    1780*   1785#
$CLR.T  023070            4799    4802#
$CMTAG  001100             897#   2437    2438    2445    2451    2452    2453
```

```
$CM1   = 000006          923#    924#    925#    926#    927#    928#    929#
$CM2   = 000014          923#    924#    925#    926#    927#    928#    929#
$CM3   = 000006          921#    923
$CM4   = 000006          929#    930#    931#    932#    933#    934#    935#
$CRLF    001215          939#   1469    1493    1551    1558    1755    1788    4794
$DBLK    004416         1891    1925    1933#
$DB20    004670         1583    1640    1661    2028#
$DOAGN   023110         4776    4797    4804    4810#
$DTBL    004406         1894    1929#
$ENDAD   023100          884    1473    2471    4806#
$ENDCT   022726         2451    4778#
$ENULL   023156         4825#
$EOP     022660         4698    4702    4741    4765#
$EOPCT   022720         2451*   4775#   4779
$ERFLG   001103          900#   1359    1399    1401    1407*   1428    1453*   1493    2287
$ERMAX   001115          906#   1401    1423*   1428    2454*
$ERROR   002460         1444#   2445
$ERRPC   001116          907#   1463*   1464*   1465    1493    1523    5683    5687    5689    5692    5694    5702    5704
                        5709    5711    5713    5714    5716    5718    5723    5725    5727    5729    5731    5734    5736
                        5738    5740    5743    5745    5747
$ERRTB   001330         1002#   1542    1546
$ERTTL   001112          904#   1462*   1493    4791    4795*
$ESCAP   001206          936#   1422*   1479    1481    1493    2453*
$FILLC   001150          919#   1758    1788
$FILLS   001147          918#   1788
$GDADR   001120          908#
$GDDAT   001124          910#
$GET42   023052         4796#
$HD    = 000000          445
$ICNT    001104          901#   1414*   1415    1417*   1427
$ILLUP   004616         1974    2005#
$ITEMB   001114          905#   1465*   1493    1521
$LF      001216          940#   1493    1788
$LOOP    023150         4818    4822#
$LPADR   001106          902#   1405*   1420*   1425    1427    2387    2467*   3206*   3414*   3574*   4205*
$LPERR   001110          903#   1405    1421*   1427    1478    2468*   2523*   2529*   2554*   2560*   2593*   2602*   2627*
                        2636*   2661*   2670*   2695*   2704*   2740*   2752*   2761*   2783*   2795*   2805*   2830*   2847*
                        2876*   2894*   2920*   2938*   2965*   2983*   3009*   3027*   3057*   3066*   3072*   3079*   3087*
                        3120*   3147*   3162*   3207*   3212*   3233*   3252*   3267*   3274*   3281*   3288*   3316*   3338*
                        3352*   3375*   3391*   3415*   3512*   3575*   3614*   3635*   3648*   3671*   3686*   3701*   3716*
                        3731*   3746*   3761*   3776*   3791*   3806*   3821*   3832*   3859*   3896*   3933*   3949*   3988*
                        3991*   4006*   4077*   4148*   4206*   4214*   4227*   4255*   4270*   4285*   4300*   4315*   4330*
                        4345*   4360*   4375*   4390*   4405*   4416*   4443*   4480*   4517*   4533*   4575*   4578*   4593*
                        4667*   4740*
$MXCNT   002456         1418    1427#
$NULL    001146          917#   1760    1788
$NWTST=  000001         2507#   2509    2535#   2537    2575#   2577    2610#   2612    2644#   2646    2678#   2680    2720#
                        2722    2764#   2766    2807#   2809    2858#   2860    2902#   2904    2947#   2949    2991#   2993
                        3035#   3037    3090#   3092    3194#   3196    3236#   3238    3292#   3294    3356#   3358    3395#
                        3397    3490#   3492    3547#   3549    3617#   3619    3652#   3654    3835#   3837    3905#   3907
                        3961#   3963    4018#   4020    4089#   4091    4192#   4194    4231#   4233    4421#   4423    4489#
                        4491    4545#   4547    4605#   4607    4679#   4681
$OCNT    004176         1821*   1850*   1863#
$OCTVL   004772         2030    2055#
$OMODE   004200         1816*   1820*   1825    1828*   1839*   1865#
$OVER    002442         1380    1398    1406    1416    1424#
```

```
$PASS   001100            898#  1412  1428  3462  4772*  4773*  4784  4824
$PWRAD  004612           2003#
$PWRDN  004470           1974#  1998  2449
$PWRMG  004606           2001#
$PWRUP  004536           1983   1988#
$QUES   001214            938#  1493
$RDCHR= ****** U         1964
$RDDEC= ****** U         1964
$RDLIN= ****** U         1964
$RDOCT= ****** U         1964
$REGAD  001152            921#
$REGO   001154            923#  1447*  5683  5692  5702  5709  5718  5734  5736  5738  5740  5743  5745
                         5747
$REG1   001156            924#  1448*  5694  5704  5711  5725  5727  5738  5743
$REG2   001160            925#  1449*  5692  5725  5727  5740  5743  5747
$REG3   001162            926#  1450*  5694  5727  5740  5745  5747
$REG4   001164            927#  1451*  5709  5740
$REG5   001166            928#  1452*
$RESRE  003506           1704#  1965
$RTRN   023146           2455   2457*  2462*  4800  4819#
$SAVRE  003450           1688#  1964
$SAVR6  004622           1982*  1988   1989*  1990*  2007#
$SCOPE  002140           1368#  2443
$SETUP= 000037            850#  1470   2443  2445  2447  2449  2451  2452  2453  2455  2467  2471  4770
$STUP = 177777            850#
$SVLAD  002414           1388   1419#
$SVPC = 000204            882#   887
$SWR  = 177400            435#   445   450   451   452   453   454   455   456   457   935   936   937
                         1360   1361  1362  1363  1364  1379  1391  1393  1394  1399  1400  1401  1408
                         1409   1410  1421  1424  1427  1435  1436  1437  1438  1439  1440  1456  1459
                         1466   1470  1476  1493  2452  2453  2455  2467  2468  2521  2553  2591  2626
                         2660   2694  2737  2781  2828  2874  2918  2963  3007  3057  3113  3210  3251
                         3313   3374  3412  3508  3560  3630  3667  3849  3923  3979  4037  4108  4209
                         4249   4435  4507  4566  4627  4698  4761  4771  4798  4811  4824
$SWRMK= 000200            435#   457   458  1364  1365  1395  1396
$TBIT   023154           2466*  4815*  4824#
$TIMES  001204            935#  1408*  1415  1418*  1427  2452*  2737*  2828*  2874*  2918*  2963*  3007*  3113*
                         3313*  3412*  3508*  3849*  4435*  4771*
$TKB    001140            914#
$TKS    001136            913#  2440
$TMP0   001170            929#  2170   2171*  2172  2173*  2596*  2630*  2664*  2698*  3510*  3527*  3531  3534*
                         5683   5736
$TMP1   001172            930#
$TMP2   001174            931#
$TMP3   001176            932#
$TMP4   001200            933#  3924*  3943  3979*  4000  4041*  4070  4112*  4141  4508*  4527  4566*  4587
                         4631*  4660   4704*  4733  5731
$TMP5   001202            934#  3115*  3163  3927*  3948  3982*  4005  4040*  4076  4111*  4147  4511*  4532
                         4569*  4592   4630*  4666  4703*  4739
$TN   = 000045            435#   445   2507  2518  2521#  2532  2535  2550  2553#  2563  2575  2588  2591#
                         2605   2610  2623  2626#  2639  2644  2657  2660#  2673  2678  2691  2694#  2707
                         2720   2734  2737#  2764  2778  2781#  2807  2825  2828#  2850  2858  2871  2874#
                         2897   2902  2915  2918#  2941  2947  2960  2963#  2986  2991  3004  3007#  3030
                         3035   3054  3057#  3090  3110  3113#  3194  3205  3210#  3236  3248  3251#  3292
                         3310   3313#  3356  3371  3374#  3395  3409  3412#  3413  3463  3490  3505  3508#
                         3541   3547  3557  3560#  3578  3617  3627  3630#  3652  3664  3667#  3835  3846
```

DEKBFB.P11 CROSS REFERENCE TABLE -- MACRO NAMES

```
AND      435#  2141  2167  2171  2194  2198  2220  2224  2228  2251  2255  2280  2284  3596  3600
COMMEN   849#
COUNT1   435#  2875  2919  2964  3008
DONE     435#  4766
ENDCOM   849#
ENTRY    435#  3205  3413  4204
ERROR    467#  2138  2146  2148  2176  2178  2203  2205  2233  2235  2260  2262  2289  2291  2329
        2331  2374  2383  2422  2533  2564  2606  2640  2674  2708  2746  2760  2789  2804  2851
        2898  2942  2987  3031  3065  3071  3078  3086  3132  3161  3226  3263  3273  3280  3287
        3326  3350  3390  3436  3458  3468  3533  3542  3607  3609  3613  3647  3681  3696  3711
        3726  3741  3756  3771  3786  3801  3816  3831  3880  3900  3945  4002  4073  4144  4226
        4265  4280  4295  4310  4325  4340  4355  4370  4385  4400  4415  4464  4484  4529  4589
        4663  4736
ESCAPE   849#
MAPADD   435#  3671  3686  3701  3716  3731  3746  3761  3776  3791  3806  3821  4255  4270  4285
        4300  4315  4330  4345  4360  4375  4390  4405
MSG1    2507#  2509
MSG10   2764#  2766
MSG11   2807#  2809
MSG12   2858#  2860
MSG13   2902#  2904
MSG14   2947#  2949
MSG15   2991#  2993
MSG16   3035#  3037
MSG17   3090#  3092
MSG2    2535#  2537
MSG20   3194#  3196
MSG21   3236#  3238
MSG22   3292#  3294
MSG23   3356#  3358
MSG24   3395#  3397
MSG25   3490#  3492
MSG26   3547#  3549
MSG27   3617#  3619
MSG3    2575#  2577
MSG30   3652#  3654
MSG31   3835#  3837
MSG32   3905#  3907
MSG33   3961#  3963
MSG34   4018#  4020
MSG35   4089#  4091
MSG36   4192#  4194
MSG37   4231#  4233
MSG4    2610#  2612
MSG40   4420#  4423
MSG41   4489#  4491
MSG42   4545#  4547
MSG43   4605#  4607
MSG44   4679#  4681
MSG5    2644#  2646
MSG6    2678#  2680
MSG7    2720#  2722
MULT     849#
NEWTST   849#  2507  2535  2575  2610  2644  2678  2720  2764  2807  2858  2902  2947  2991  3035
        3090  3194  3236  3292  3356  3395  3490  3547  3617  3652  3835  3905  3961  4018  4089
        4192  4231  4421  4489  4545  4605  4679
```

DEKBFB.P11 CROSS REFERENCE TABLE -- MACRO NAMES

```
POP      849#  1709  1920  1992
PUSH     849#  1689  1879  1976
SAVTST   435#  1445
SCOPE    468#  2518  2550  2588  2623  2657  2691  2734  2778  2825  2871  2915  2960  3004  3054
        3110  3164  3248  3310  3371  3409  3505  3557  3627  3664  3846  3920  3976  4034  4105
        4166  4246  4432  4504  4563  4624  4697  4766
SETTRA  1950#  1960  1961  1962  1963  1964  1965  1966  1967
SETUP    849#  2436
SKIP     849#  2532  2563  2605  2639  2673  2707  2850  2897  2941  2986  3030  3463  3541  3578
        3950  4007  4039  4078  4110  4149  4534  4594  4629  4668  4702  4741
SLASH    849#
SPACE    849#
SSCOPE   435#  1369  3133
STARS    849#   862   888   986  1353  1428  1670  1716  1788  1866  1934  1969  2016  2507  2516
        2535  2548  2575  2586  2610  2621  2644  2655  2678  2689  2720  2732  2764  2776  2807
        2823  2858  2869  2902  2913  2947  2958  2991  3002  3035  3052  3090  3108  3194  3204
        3236  3246  3292  3308  3356  3369  3395  3407  3490  3503  3547  3555  3617  3625  3652
        3662  3835  3844  3905  3918  3961  3974  4018  4032  4089  4103  4192  4203  4231  4244
        4421  4430  4489  4502  4545  4561  4605  4622  4679  4696  4753
TRACK    435#  2517  2549  2587  2622  2656  2690  2733  2777  2824  2870  2914  2959  3003  3053
        3109  3247  3309  3370  3408  3504  3556  3626  3663  3845  3919  3975  4033  4104  4245
        4431  4503  4562  4623
TRACK1   435#  2518  2550  2588  2623  2657  2691  2734  2778  2825  2871  2915  2960  3004  3054
        3110  3248  3310  3371  3409  3505  3557  3627  3664  3846  3920  3976  4034  4105  4246
        4432  4504  4563  4624
TRMTRP  1950#
TYPBIN   849#
TYPDEC   849#  4784  4791
TYPNAM   849#  2469
TYPNUM   849#
TYPOCS   849#
TYPOCT   849#
TYPTXT   849#  4780  4787
USER     435#   941
ZEROER   435#  1483
$$CMRE   888#   923   924   925   926   927   928
$$CMTM   888#   929   930   931   932   933   934
$$ESCA   849#
$$NEWT   849#  2507  2535  2575  2610  2644  2678  2720  2764  2807  2858  2902  2947  2991  3035
        3090  3194  3236  3292  3356  3395  3490  3547  3617  3652  3835  3905  3961  4018  4089
        4192  4231  4421  4489  4545  4605  4679
$$SET   1950#  1960  1961  1962  1963  1964  1965  1966  1967
$$SETU  2455#
$$SKIP   849#  2532  2563  2605  2639  2673  2707  2850  2897  2941  2986  3030  3463  3541  3578
        3950  4007  4039  4078  4534  4594  4629  4668
.HEADE   435#
.SETUP   435#   850
.SWRHI   435#   445
.SWRLO   458#
.$ACTI   435#   862
.$CATC   435#   850
.$CMTA   435#   888
.$DB2O   435#  2016
.$EOP    435#  4753
.$ERRO   435#  1428
.$POWE   435#  1969
```

```
.$SAVE     4358    1670
.$SCOP     4358    1353
.$TRAP     4358    1934
.$TYPD     4358    1966
.$TYPE     4358    1716
.$TYPO     4358    1788
.1170      4358     459
```

DEKBFB.P11 CROSS REFERENCE TABLE -- PERMANENT SYMBOLS

```
        4567  4573  4576  4585  4632  4635  4645  4705  4708  4719  4767  4768  4770  4771  4795
        4798
CLRB    1407  1780  1919  4062  4133  4652  4725
CMP     1389  1415  1473  1911  2110  2136  2327  2440  2471  2527  2558  2600  2634  2668  2702
        2744  2756  2787  2792  2799  2802  2838  2842  2845  2883  2888  2890  2927  2932  2934
        2972  2977  2979  3016  3021  3023  3076  3084  3124  3128  3154  3221  3231  3261  3271
        3278  3285  3324  3332  3334  3343  3388  3422  3431  3434  3441  3445  3450  3454  3456
        3464  3466  3521  3523  3529  3538  3569  3576  3586  3592  3645  3679  3694  3709  3724
        3739  3754  3769  3784  3799  3814  3829  3873  3877  3891  3897  3943  4000  4070  4141
        4224  4263  4278  4293  4308  4323  4338  4353  4368  4383  4398  4413  4457  4461  4475
        4481  4527  4587  4660  4733
CMPB    1397  1401  1573  1579  1589  1750  1752  1758  1778  1782
COM     2141  2143  2167  2169  2171  2173  2194  2196  2198  2200  2220  2222  2224  2226  2228
        2230  2251  2253  2255  2257  2280  2282  2284  2286  3597  3599  3600  3602  4815
DEC     1526  2038  4774
DECB    1762  1839  1850
EMT      467
HALT     857  1458  1472  1475  1740  1984  2005  2127  2314  2356  2411
INC     1414  1446  1462  1601  1845  1853  1897  1990  2041  2124  2311  2353  2386  2408  2469
        2791  2801  3127  3346  3448  3452  3882  3890  3893  4067  4138  4466  4474  4477  4657
        4730  4772
INCB    1419  1453  1784
IOT      468
JMP      861  2179  2206  2236  2263  2292  3327  3437  4823
JSR     1468  1583  1594  1599  1640  1661  1757  1764  1770  2598  2632  2666  2700  2832  2840
        2885  2929  2974  3018  3126  3156  3314  3345  3509  3525  4806
MOV     1375  1376  1377  1378  1384  1385  1387  1390  1395  1405  1417  1418  1420  1421  1424
        1425  1447  1448  1449  1450  1451  1452  1455  1463  1478  1481  1485  1486  1488  1489
        1490  1491  1519  1523  1547  1554  1563  1564  1566  1570  1576  1582  1585  1593  1598
        1606  1607  1624  1632  1637  1638  1639  1642  1647  1655  1658  1659  1660  1663  1666
        1689  1690  1691  1692  1693  1694  1695  1696  1697  1698  1705  1706  1707  1708  1709
        1710  1711  1712  1713  1714  1742  1743  1747  1760  1814  1822  1823  1824  1830  1837
        1855  1856  1857  1858  1859  1880  1881  1882  1883  1884  1885  1886  1891  1894  1914
        1920  1921  1922  1923  1924  1926  1927  1943  1944  1947  1974  1975  1976  1977  1978
        1979  1980  1981  1982  1983  1988  1992  1993  1994  1995  1996  1997  1998  1999  2002
        2029  2030  2031  2032  2033  2034  2037  2042  2075  2092  2108  2132  2133  2134  2135
        2149  2150  2151  2165  2192  2218  2249  2278  2319  2320  2321  2323  2324  2332  2333
        2334  2335  2361  2363  2364  2365  2366  2367  2368  2369  2370  2373  2376  2377  2381
        2382  2387  2389  2391  2392  2393  2394  2416  2417  2418  2419  2420  2421  2424  2425
        2426  2437  2438  2442  2443  2444  2445  2446  2447  2448  2449  2450  2451  2455  2456
        2457  2458  2460  2462  2465  2467  2468  2477  2479  2480  2482  2483  2484  2485  2492
        2493  2494  2495  2496  2497  2498  2519  2521  2522  2523  2525  2529  2551  2553  2554
        2556  2560  2589  2591  2592  2593  2596  2602  2624  2626  2627  2630  2636  2658  2660
        2661  2664  2670  2692  2694  2695  2698  2704  2735  2737  2740  2742  2743  2752  2754
        2755  2761  2779  2783  2785  2786  2795  2797  2798  2805  2826  2828  2829  2830  2831
        2835  2836  2847  2872  2874  2876  2877  2879  2881  2882  2894  2916  2918  2920  2921
        2923  2925  2926  2938  2961  2963  2965  2966  2968  2970  2971  2983  3005  3007  3009
        3010  3012  3014  3015  3027  3055  3057  3063  3066  3069  3072  3073  3075  3079  3080
        3082  3087  3111  3113  3115  3118  3120  3122  3123  3139  3140  3141  3142  3143  3144
        3145  3147  3148  3150  3151  3162  3163  3173  3175  3176  3177  3178  3179  3180  3181
        3182  3183  3184  3185  3186  3187  3188  3189  3190  3191  3192  3206  3207  3208  3209
        3211  3212  3213  3214  3215  3216  3217  3220  3223  3228  3233  3249  3251  3252  3253
        3254  3255  3256  3259  3267  3268  3270  3274  3275  3277  3281  3282  3284  3288  3311
        3313  3315  3316  3317  3320  3328  3330  3331  3338  3339  3341  3342  3352  3372  3374
        3375  3376  3379  3380  3383  3387  3391  3410  3412  3414  3415  3416  3417  3418  3419
        3420  3425  3426  3427  3428  3430  3438  3439  3444  3449  3473  3474  3481  3483  3484
```

```
                3486  3506  3508  3511  3512  3513  3514  3515  3516  3517  3519  3520  3527  3558  3561
                3562  3563  3564  3565  3574  3575  3579  3582  3583  3585  3614  3628  3631  3632  3633
                3634  3635  3638  3648  3665  3671  3673  3674  3675  3677  3678  3686  3688  3689  3690
                3692  3693  3701  3703  3704  3705  3707  3708  3716  3718  3719  3720  3722  3723  3731
                3733  3734  3735  3737  3738  3746  3748  3749  3750  3752  3753  3761  3763  3764  3765
                3767  3768  3776  3778  3779  3780  3782  3783  3791  3793  3794  3795  3797  3798  3806
                3808  3809  3810  3812  3813  3821  3823  3824  3825  3827  3828  3832  3847  3849  3852
                3854  3855  3856  3857  3858  3859  3860  3862  3871  3872  3876  3885  3896  3921  3924
                3926  3927  3928  3929  3930  3932  3933  3934  3935  3936  3937  3939  3940  3946  3947
                3948  3949  3953  3954  3955  3956  3957  3977  3979  3981  3982  3983  3984  3985  3987
                3988  3990  3991  3992  3993  3994  3996  3997  4003  4004  4005  4006  4010  4011  4012
                4013  4014  4035  4037  4040  4041  4043  4044  4046  4047  4048  4049  4053  4054  4056
                4057  4060  4061  4064  4072  4074  4075  4076  4077  4081  4082  4083  4084  4085  4106
                4108  4111  4112  4114  4115  4117  4118  4119  4120  4124  4125  4127  4128  4131  4132
                4135  4143  4145  4146  4147  4148  4152  4153  4154  4155  4156  4179  4182  4185  4187
                4189  4205  4206  4207  4208  4209  4210  4211  4212  4213  4214  4217  4227  4247  4249
                4255  4257  4258  4259  4261  4262  4270  4272  4273  4274  4276  4277  4285  4287  4288
                4289  4291  4292  4300  4302  4303  4304  4306  4307  4315  4317  4318  4319  4321  4322
                4330  4332  4333  4334  4336  4337  4345  4347  4348  4349  4351  4352  4360  4362  4363
                4364  4366  4367  4375  4377  4378  4379  4381  4382  4390  4392  4393  4394  4396  4397
                4405  4407  4408  4409  4411  4412  4416  4433  4435  4436  4438  4439  4440  4441  4442
                4443  4444  4446  4455  4456  4460  4469  4480  4505  4508  4510  4511  4512  4513  4514
                4516  4517  4518  4519  4520  4521  4523  4524  4530  4531  4532  4533  4537  4538  4539
                4540  4541  4564  4566  4568  4569  4570  4571  4572  4574  4575  4577  4578  4579  4580
                4581  4583  4584  4590  4591  4592  4593  4597  4598  4599  4600  4601  4625  4627  4630
                4631  4633  4634  4636  4637  4638  4639  4643  4644  4646  4647  4650  4651  4654  4662
                4664  4665  4666  4667  4671  4672  4673  4674  4675  4698  4700  4703  4704  4706  4707
                4709  4710  4711  4712  4716  4717  4719  4720  4723  4724  4727  4735  4737  4738  4739
                4740  4744  4745  4746  4747  4748  4777  4784  4791  4796  4799  4803  4811  4818
MOVB            1423  1445  1465  1521  1744  1769  1777  1815  1816  1819  1820  1821  1825  1828  1829
                1848  1889  1892  1906  1909  1918  1946  2036  2454
NEG             1826  1888
NOP             2524  2555  2594  2628  2662  2696  2741  2753  2784  2796  2834  2880  2924  2969  3013
                3061  3067  3074  3081  3121  3149  3219  3258  3269  3276  3283  3319  3340  3382  3584
                3637  3676  3691  3706  3721  3736  3751  3766  3781  3796  3811  3826  3875  3938  3995
                4216  4260  4275  4290  4305  4320  4335  4350  4365  4380  4395  4410  4459  4522  4582
                4807  4808  4809
RESET           4805
ROL             1379  1832  1834  1835  1836  1838
ROR             1628  1629  1630  2046  2047  2048  2049  2050  2051
RTI             1426  1492  1699  1715  1749  1860  1928  2004  2457  3958  4015  4086  4157  4542  4602
                4676  4749  4819
RTS             1608  1648  1667  1786  1948  2044  2112
RTT             2077  2095  2152  2336  2395  2427  2461  2462
SOB             3567  3590
SUB             1464  1895  2841
SXT             4063  4134  4653  4726
TRAP            1950  1960  1961  1962  1963  1964  1965  1966  1967
TST             1386  1412  1456  1470  1479  1603  1746  1754  1773  1843  1900  1910  1945  2144  2174
                2201  2231  2258  2325  2380  2384  2531  2562  2604  2638  2672  2706  2849  2896  2940
                2985  3029  3130  3159  3224  3321  3348  3462  3531  3540  3605  3611  3869  4453
TSTB            1399  1567  1738  1775  1902  1916  2287
XOR             2379  3218  3229  3257  3264
.ASCII           938   939  4972  4987  5003  5021  5072  5086  5121  5141  5159  5168  5177  5259  5275
                5290  5306  5321  5412  5428  5441  5451  5467  5476  5490  5514  5564  5573  5582  5599
                5626  5652  5661
```

DEKBFB.P11 CROSS REFERENCE TABLE -- PERMANENT SYMBOLS

```
             4007  4018  4020  4032  4037  4039  4078  4089  4091  4103  4108  4110  4149  4192  4194
             4203  4209  4231  4233  4244  4249  4421  4423  4430  4435  4436  4489  4491  4502  4507
             4534  4545  4547  4561  4566  4594  4605  4607  4622  4627  4629  4668  4679  4681  4696
             4698  4702  4741  4753  4757  4758  4759  4761  4763  4764  4766  4775  4778  4780  4782
             4789  4798  4803  4811  4823  4824  4825
.IFF          454   456   457   458   863   885   887   889   895   921   987  1354  1392  1395  1401
             1427  1429  1434  1459  1470  1483  1671  1717  1789  1867  1935  1944  1970  2000  2017
             2442  2470  2472  2507  2508  2509  2517  2521  2533  2535  2536  2537  2549  2553  2564
             2575  2576  2577  2587  2591  2606  2610  2611  2612  2622  2626  2640  2644  2645  2646
             2656  2660  2674  2678  2679  2680  2690  2694  2708  2720  2721  2722  2733  2737  2764
             2765  2766  2777  2781  2807  2808  2809  2824  2828  2851  2858  2859  2860  2870  2874
             2898  2902  2903  2904  2914  2918  2942  2947  2948  2949  2959  2963  2987  2991  2992
             2993  3003  3007  3031  3035  3036  3037  3053  3057  3090  3091  3092  3109  3113  3194
             3195  3196  3205  3210  3236  3237  3238  3247  3251  3292  3293  3294  3309  3313  3356
             3357  3358  3370  3374  3395  3396  3397  3408  3413  3464  3490  3491  3492  3504  3508
             3542  3547  3548  3549  3556  3560  3579  3617  3618  3619  3626  3630  3652  3653  3654
             3663  3667  3835  3836  3837  3845  3849  3905  3906  3907  3919  3923  3951  3961  3962
             3963  3975  3979  4008  4018  4019  4020  4033  4037  4040  4079  4089  4090  4091  4104
             4108  4110  4149  4192  4193  4194  4204  4209  4231  4232  4233  4245  4249  4421  4422
             4423  4431  4435  4489  4490  4491  4503  4507  4535  4545  4546  4547  4562  4566  4595
             4605  4606  4607  4623  4627  4630  4669  4680  4681  4697  4698  4702  4741  4754  4763
             4766  4775  4778  4824
.IFT         1409  1469  2476  4783  4790
.IFTF        1407  1468  2476  4783  4790
.IIF          435   440   445   450   451   452   454   457   458   459   857   941  1360  1361  1362
             1363  1364  1365  1408  1409  1424  1427  1428  1435  1436  1437  1438  1439  1440  1470
             1493  1788  1959  1960  1961  1962  1963  1964  1965  1966  1967  2443  2445  2451  2452
             2453  2467  2468  2471  4759  4770  4771  4785  4792  4824  4826
.IRP          850   941  1369  1445  1483  1689  1709  1880  1920  1976  1992  2507  2535  2575  2610
             2644  2678  2720  2764  2807  2858  2902  2947  2991  3035  3090  3194  3236  3292  3356
             3395  3490  3547  3617  3652  3835  3905  3961  4018  4089  4192  4231  4421  4489  4545
             4605  4679  4766
.LIST         435   457   849   850   857   921   923   924   925   926   927   928   929   930   931
              932   933   934   935  1364  1470  1949  1950  1959  1960  1961  1962  1963  1964  1965
             1966  1967  1968  2471  2476  2507  2521  2535  2553  2575  2591  2610  2626  2644  2660
             2678  2694  2720  2737  2764  2781  2807  2828  2858  2874  2902  2918  2947  2963  2991
             3007  3035  3057  3090  3113  3194  3210  3236  3251  3292  3313  3356  3374  3395  3412
             3490  3508  3547  3560  3617  3630  3652  3667  3835  3849  3905  3923  3961  3979  4018
             4037  4089  4108  4192  4209  4231  4249  4421  4435  4489  4507  4545  4566  4605  4627
             4679  4698  4770  4783  4790  4798
.MACRO        435   458   888  1950  2455  2507  2535  2575  2610  2644  2678  2720  2764  2807  2858
             2902  2947  2991  3035  3090  3194  3236  3292  3356  3395  3490  3547  3617  3652  3835
             3905  3961  4018  4089  4192  4231  4420  4489  4545  4605  4679
.MCALL        435   849
.NLIST        435   457   849   850   857   921   923   924   925   926   927   928   929   930   931
              932   933   934   935  1364  1470  1949  1950  1959  1960  1961  1962  1963  1964  1965
             1966  1967  1968  2471  2476  2507  2521  2535  2553  2575  2591  2610  2626  2644  2660
             2678  2694  2720  2737  2764  2781  2807  2828  2858  2874  2902  2918  2947  2963  2991
             3007  3035  3057  3090  3113  3194  3210  3236  3251  3292  3313  3356  3374  3395  3412
             3490  3508  3547  3560  3617  3630  3652  3667  3835  3849  3905  3923  3961  3979  4018
             4037  4089  4108  4192  4209  4231  4249  4421  4435  4489  4507  4545  4566  4605  4627
             4679  4698  4770  4783  4790  4798
.PAGE         888   986
.REM            1
.REPT         857   923   929
.SBTTL        446   460   585   596   610   759   851   858   864   890   988  1355  1430  1495  1613
```

```
            1672    1718    1790    1868    1936    1951    1971    2018    2058    2062    2081    2100    2114    2154    2181
            2208    2239    2266    2295    2297    2340    2399    2432    2433    2507    2535    2575    2610    2644    2678
            2720    2764    2807    2858    2902    2947    2991    3035    3090    3194    3236    3292    3356    3395    3490
            3547    3617    3652    3835    3905    3961    4018    4089    4161    4162    4163    4192    4231    4[illegible]    4489
            4545    4605    4679    4755    4829
.TITLE       435
.WORD        857     884     886     898     901     902     903     904     907     908     909     910     911     912     921
             923     924     925     926     927     928     929     930     931     932     933     934     941     943     945
             946     947     948     949     950     951     953     955     957     959     961     963     964     965     966
             967     968     969     970     971     972     973     974     975     976     977     978     979     980     981
             982     983     985    1550    1557    1587    1644    1665    1785    1865    2001    2003    2125    2312    2354
            2409    3883    4068    4139    4467    4658    4731    4775    4778    4824    5676    5678    5680    5683    5685
            5687    5689    5692    5694    5696    5699    5702    5704    5706    5709    5711    5713    5714    5716    5718
            5720    5723    5725    5727    5729    5731    5734    5736    5738    5740    5743    5745    5747
```

ERRORS DETECTED: 0
DEFAULT GLOBALS GENERATED: 0

*,DEKBFB.SEQ/SOL/CRF/PAGNUM=DEKBFB
RUN-TIME: 57 56 9 SECONDS
RUN-TIME RATIO: 243/124=1.9
CORE USED: 27K (53 PAGES)

Spooler runtime 22 Seconds, 95 KCS, 634 disk reads, 4 disk writes, 137 pages