11/70-11/74 11/70 UNIBUS MAP AH-7980D-MC NOV 1980
CEKBFDO FICHE 1 OF 1 COPYRIGHT@ 75-80 MADE IN USA

IDENTIFICATION

PRODUCT CODE: AC-7979D-MC

PRODUCT NAME: CEKBFDO 11/70 UNIBUS MAP

DATE CREATED: MAY, 1980

MAINTAINER: DIAGNOSTIC ENGINEERING

TABLE OF CONTENTS

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. THIS PROGRAM ALSO PROVIDES DIAGNOSTIC COVERAGE FOR THE 'CACHE BYPASS' FUNCTION ON MAP PAGE FOR KB11-CM AND SUBSEQUENT PROCESSORS.

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 (17777770) 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.

REQUIREMENTS

EQUIPMENT

THE BASIC PDP-11/70 COMPUTER, INCLUDING THE CPU, CACHE, MEMORY MANAGEMENT, AND AN LA-30 OR EQUIVALENT DEVICE FOR ERROR MESSAGES.
NOTE: THIS DIAGNOSTIC SUPPORTS THE PDP-11/74, AN EXPERIMENTAL, IN-HOUSE PROCESSOR.

STORAGE

THIS PROGRAM WILL REQUIRE 8K TO LOAD BUT WILL UTILIZE ALL EXISTING CORE FOR A DUAL ADDRESSING TEST OF MEMORY 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 WIL. 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 SPFCIAL 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
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 02 01 00

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

A2. OFFSET                                               12 11 10 09 08 07 06 05 04 03 02 01 00
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 02 01 00

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 02 01 00
D.  MAP REG.[NO.]    +  21 20 19 18 17 16 15 14 13 12 11 10 09 08 07 06 05 04 03 02 01
                        ------------------------  ------------------------------------------
E.  PHYSICAL ADDR.      21 20 19 18 17 16 15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00
```

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 GETS 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 OUT ON UNIBUS
- C2: IF MAP RELOCATION IS ENABLED THEN BITS <17:13> SELECT ONE OF THE 36 (OCTAL) MAP REGISTERS.
- C3: LOWER 13 BITS OF UNIBUS ADDRESS, ADDED TO SELECTED MAP REGISTER

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

E: PHYSICAL ADDRESS GENERATED BY UNIBUS MAP AND SENT TO THE CACHE.

9. REVISION HISTORY

REV DO TESTS 27 AND 36 WILL BE BYPASSED ON SYSTEMS WITH GREATER THAN 1920K OF MEMORY.

@
.END

```
                        .TITLE  CEKBFDO 11/70 UNIBUS MAP
                        ;*COPYRIGHT (C) 1975,1980
                        ;*DIGITAL EQUIPMENT CORP.
                        ;*MAYNARD, MASS. 01754
                        ;*
                        ;*
                        ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
                        ;*PACKAGE (MAINDEC-11-DZQAC-A5).
                        ;*

                        .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
000007                  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
000006                  SP=%6                   ;;STACK POINTER
                        .EQUIV  SP,KSP          ;;KERNEL STACK POINTER
                        .EQUIV  SP,SSP          ;;SUPERVISOR STACK POINTER
                        .EQUIV  SP,USP          ;;USER STACK POINTER
000007                  PC=%7                   ;;PROGRAM COUNTER

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

                        ;*''SWITCH REGISTER'' SWITCH DEFINITIONS
100000                  SW15=   100000
040000                  SW14=   40000
020000                  SW13=   20000
010000                  SW12=   10000
004000                  SW11=   4000
002000                  SW10=   2000
001000                  SW09=   1000
000400                  SW08=   400
000200                  SW07=   200
000100                  SW06=   100
000040                  SW05=   40
000020                  SW04=   20
000010                  SW03=   10
000004                  SW02=   4
000002                  SW01=   2
000001                  SW00=   1
                        .EQUIV  SW09,SW9
                        .EQUIV  SW08,SW8
                        .EQUIV  SW07,SW7
                        .EQUIV  SW06,SW6
                        .EQUIV  SW05,SW5
                        .EQUIV  SW04,SW4
                        .EQUIV  SW03,SW3
                        .EQUIV  SW02,SW2
                        .EQUIV  SW01,SW1
                        .EQUIV  SW00,SW0

                        ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
100000                  BIT15=  100000
040000                  BIT14=  40000
020000                  BIT13=  20000
010000                  BIT12=  10000
004000                  BIT11=  4000
002000                  BIT10=  2000
```

```
                        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
```

```
                        UDPAR1= 177662
                        UDPAR2= 177664
                        UDPAR3= 177666
                        UDPAR4= 177670
                        UDPAR5= 177672
                        UDPAR6= 177674
                        UDPAR7= 177676

                        ;*SUPERVISOR ''I'' PAGE DESCRIPTOR REGISTERS

                        SIPDR0= 172200
                        SIPDR1= 172202
                        SIPDR2= 172204
                        SIPDR3= 172206
                        SIPDR4= 172210
                        SIPDR5= 172212
                        SIPDR6= 172214
                        SIPDR7= 172216

                        ;*SUPERVISOR 'D'' PAGE DESCRIPTOR REGISTERS

                        SDPDR0= 172220
                        SDPDR1= 172222
                        SDPDR2= 172224
                        SDPDR3= 172226
                        SDPDR4= 172230
                        SDPDR5= 172232
                        SDPDR6= 172234
                        SDPDR7= 172236

                        ;*SUPERVISOR ''I'' PAGE ADDRESS REGISTERS

                        SIPAR0= 172240
                        SIPAR1= 172242
                        SIPAR2= 172244
                        SIPAR3= 172246
                        SIPAR4= 172250
                        SIPAR5= 172252
                        SIPAR6= 172254
                        SIPAR7= 172256

                        ;*SUPERVISOR 'D'' PAGE ADDRESS REGISTERS

                        SDPAR0= 172260
                        SDPAR1= 172262
                        SDPAR2= 172264
                        SDPAR3= 172266
                        SDPAR4= 172270
                        SDPAR5= 172272
                        SDPAR6= 172274
                        SDPAR7= 172276

                        ;*KERNEL ''I'' PAGE DESCRIPTOR REGISTERS

                        KIPDR0= 172300
                        KIPDR1= 172302
```

```
                        ;*KERNEL 'D' PAGE DESCRIPTOR REGISTERS

                        ;*KERNEL "I" PAGE ADDRESS REGISTERS

                        ;*KERNEL 'D' PAGE ADDRESS REGISTERS

                        .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'
```

```
337     170214          MAPL03 = 170214
338     170216          MAPH03 = 170216
339     170220          MAPL04 = 170220
340     170222          MAPH04 = 170222
341     170224          MAPL05 = 170224
342     170226          MAPH05 = 170226
343     170230          MAPL06   170230
344     170232          MAPH06 = 170232
345     170234          MAPL07 = 170234
346     170236          MAPH07 = 170236
347     170240          MAPL10 = 170240
348     170242          MAPH10 = 170242
349     170244          MAPL11 = 170244
350     170246          MAPH11 = 170246
351     170250          MAPL12 = 170250
352     170252          MAPH12 = 170252
353     170254          MAPL13 = 170254
354     170256          MAPH13 = 170256
355     170260          MAPL14 = 170260
356     170262          MAPH14 = 170262
357     170264          MAPL15 = 170264
358     170266          MAPH15 = 170266
359     170270          MAPL16 = 170270
360     170272          MAPH16 = 170272
361     170274          MAPL17 = 170274
362     170276          MAPH17 = 170276
363     170300          MAPL20 = 170300
364     170302          MAPH20 = 170302
365     170304          MAPL21 = 170304
366     170306          MAPH21 = 170306
367     170310          MAPL22 = 170310
368     170312          MAPH22 = 170312
369     170314          MAPL23 = 170314
370     170316          MAPH23 = 170316
371     170320          MAPL24 = 170320
372     170320          MAPH24 = 170320
373     170324          MAPL25 = 170324
374     170326          MAPH25 = 170326
375     170330          MAPL26 = 170330
376     170332          MAPH26 = 170332
377     170334          MAPL27 = 170334
378     170336          MAPH27 = 170336
379     170340          MAPL30 = 170340
380     170342          MAPH30 = 170342
381     170344          MAPL31 = 170344
382     170346          MAPH31 = 170346
383     170350          MAPL32 = 170350
384     170352          MAPH32 = 170352
385     170354          MAPL33 = 170354
386     170356          MAPH33 = 170356
387     170360          MAPL34 = 170360
388     170362          MAPH34 = 170362
389     170364          MAPL35 = 170364
390     170366          MAPH35 = 170366
391     170370          MAPL36 = 170370
392     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


                        100000                  BYP=BIT15
                        010000                  VCIP=BIT12
                        000040                  S1=BIT5
                        000020                  S0=BIT4
                        000054                  S1MOM1=BIT5+BIT3+BIT2
                        000034                  S0MOM1=BIT4+BIT3+BIT2
                        177744                  MSER=177744

                                        .SBTTL  TRAP CATCHER

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

                                        .SBTTL  STARTING ADDRESS(ES)
                                                .=200
                        000200
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

                                $SVPC=.                 ;;SAVE LOCATION COUNTER
000204                          .=46                    ;;SET LOCATION COUNTER
000046                          .WORD   $ENDAD          ;;SET LOC.46 TO ADDRESS $ENDAD
000046  024600                  .=52                    ;;SET LOCATION COUNTER
000052                          .WORD   0               ;;SET LOC.52 TO ZERO
000052  000000                  .=$SVPC                 ;; RESTORE LOCATION COUNTER
000204
```

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

                                .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          $REGAD: .WORD   0               ;;CONTAINS THE  FROM
                                                        ;;WHICH  ($REG0) WAS OBTAINED
001154  000000          $REG0:  .WORD   0               ;;CONTAINS (($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
```

```
                                                ;ADDRESS GENERATED FOR TYPE OUT
                                                ;LOGICAL AND OF FAILING ADDRESSES
                                                ;LOGICAL OR OF FAILING ADDRESSES
                                                ;LOGICAL AND OF BAD DATA
                                                ;LOGICAL OR OF BAD DATA
                                                ;LOGICAL AND OF PATTERN LOADED
                                                ;LOGICAL OR OF PATTERN LOADED
                                                ;HOLDS NUMBER TO PUT IN PAR TO CAUSE THE
                                                ;LOWEST USEABLE MAP REGISTER TO RESPOND
                                                ;HOLDS NUMBER TO PUT IN PAR TO CAUSE THE
                                                ;HIGHEST USEABLE MAP REGISTER TO RESPOND
                                                ;HOLDS I/O PAGE ADDR OF LOW 16 BITS OF
                                                ;THE LOWEST USEABLE MAP REGISTER
                                                ;HOLDS I/O PAGE ADDR OF HIGH 16 BITS OF
                                                ;OF THE LOWEST USEABLE MAP REGISTER
                                                ;HOLDS I/O PAGE ADDR OF LOW 16 BITS OF
                                                ;THE HIGHEST USEABLE MAP REGISTER
                                                ;HOLDS I/O PAGE ADDR OF HIGH 16 BITS OF
                                                ;THE HIGHEST USEABLE MAP REGISTER
                                                ;MULTIPLE ERROR ERROR COUNTER
                                                ;AUXILIARY COUNTER
                                                ;FLAG TO INDICATE TO LAST PROGRAM PASS N
                                                ;HOLDS TEST NUMBER FOR ERROR TYPE OUTS
                                                ;HOLDS THE EXPECTED CPU ERROR CODE
                                                ;HOLDS RECEIVED CPU ERROR CONDITION
                                                ;HOLDS LOWER 16 BITS OF CACHE ADDR
                                                ;HOLDS UPPER 6 BITS OF CACHE ADDR
                                                ;HOLDS RECEIVED PARITY ERROR CONDITION
                                                ;HOLDS CONTENTS OF CONTROL REGISTER
                                                ;HOLDS CONTENTS OF MAINTENENCE REGISTER
                                                ;HOLDS PC OF INST THAT CAUSED TRAP
                                                ;HOLDS THE RETURN ADDRESS AFTER A TRAP
                                                ;HOLDS THE OLD PROCESSOR STATUS
                                                ;HOLDS OLD PSW FOR TBITRESTORE
                                                ;HOLDS CONTENTS OF MMR0 AFTER TRAP
                                                ;HOLDS CONTENTS OF MMR1 AFTER TRAP
                                                ;HOLDS CONTENTS OF MMR2 AFTER TRAP
                                                ;WILL HOLD P.A.R. DATA FOR TOP OF MEMORY
                                                ;RETRY FLAG IN CASE OF PARITY ABORTS
                                                ;LOCATION TO HOLD ESCAPE ADDRESS ON
                                                ;PARITY ERRORS.
                                                ;PATTERN TO BE USED TO LOAD INTO MEMORY
                                                ;1174 WITHOUT MP CACHE FLAG
                                                ;1174 WITH MP CACHE FLAG
                                                ;KB11CM FLAG (1170 WITH MP MODS)
                                                ;CISP OPTION PRESENT FLAG

                                                ;OPCODE FOR MFPT INSTRUCTION (AVAILABLE ON KB11-E AND KB11-EM ONLY)
                                                        MFPT=7
```

516						
517	001224	000000	ADRAND:	.WORD	0	
518	001226	000000	ADDRCR:	.WORD	0	
519	001230	000000	DATAND:	.WORD	0	
520	001232	000000	DATAOR:	.WORD	0	
521	001234	000000	PATAND:	.WORD	0	
522	001236	000000	PATTOR:	.WORD	0	
523	001240	000000	LOWEST:	.WORD	0	
524						
525	001242	000000	HIGEST:	.WORD	0	
526						
527	001244	000000	LREGL:	.WORD	0	
528						
529	001246	000000	LREGU:	.WORD	0	
530						
531	001250	000000	HREGL:	.WORD	0	
532						
533	001252	000000	HREGU:	.WORD	0	
534						
535	001254	000000	ERRCNT:	.WORD	0	
536	001256	000000	CNTR:	.WORD	0	
537	001260	000000	FLAG:	.WORD	0	
538	001262	000000	TESTNO:	.WORD	0	
539	001264	000000	CPUEXP:	.WORD	0	
540	001266	000000	PCPUER:	.WORD	0	
541	001270	000000	PLOADR:	.WORD	0	
542	001272	000000	PHIADR:	.WORD	0	
543	001274	000000	PPARER:	.WORD	0	
544	001276	000000	PCONTR:	.WORD	0	
545	001300	000000	PMAINT:	.WORD	0	
546	001302	000000	BADPC:	.WORD	0	
547	001304	000000	OLDPC:	.WORD	0	
548	001306	000000	OLDPS:	.WORD	0	
549	001310	000000	OLDPSW:	.WORD	0	
550	001312	000000	PMMR0:	.WORD	0	
551	001314	000000	PMMR1:	.WORD	0	
552	001316	000000	PMMR2:	.WORD	0	
553	001320	000000	RSIZE:	.WORD	0	
554	001322	000000	RETRY:	.WORD	0	
555	001324	000000	NXTTST:	.WORD	0	
556						
557	001326	000200	DATA:	.WORD	200	
558	001330	000	KB11E:	.BYTE	0	;1174 WITHOUT MP CACHE FLAG
559	001331	000	KB11EM:	.BYTE	0	;1174 WITH MP CACHE FLAG
560	001332	000	KB11CM:	.BYTE	0	;KB11CM FLAG (1170 WITH MP MODS)
561	001333	000	CISP:	.BYTE	0	;CISP OPTION PRESENT FLAG
562						
563						
564		000007		MFPT=7		

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

                                .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


001334                          $ERRTB:
                                ;ITEM1
001334  025005                  EM1                     ;NOT THE CORRECT CPU TRAP CONDITION THRU ERRVEC (#004)
001336  033627                  DH1                     ;RECEIVD EXPECTD TESTNO  PC AT ABORT
001340  036764                  DT1                     ;PCPUER,CPUEXP,TESTNO,BADPC,0
001342  037606                  DF1                     ; 0, 0, 0, 0

                                ;ITEM 2
001344  025067                  EM2                     ;UNEXPECTED CPU TRAP THRU ERRVEC (#004)
001346  033673                  DH2                     ;RECEIVD TESTNO  PC AT ABORT
001350  036776                  DT2                     ;PCPUER,TESTNO,BADPC
001352  037612                  DF2                     ; 0, 0, 0

                                ;ITEM 3
001354  025136                  EM3                     ;UNEXPECTED CACHE PARITY ERROR THRU CACHVEC (#114)
                                                        ;WILL RETRY TEST ONCE
001356  033727                  DH3                     ;PARITY  ADDRESS   MAINTEN CONTROL
                                                        ;CONDITN REFERENCD REGISTR REGISTR TESTNO PC AT ABORT
001360  037006                  DT3                     ;PPARER,LOADDR,PMAINT,PCONTR,TESTNO,BADPC,0
001362  037615                  DF3                     ; 0, 2, 0, 0, 0, 0

                                ;ITEM 4
001364  025237                  EM4                     ;UNEXPECTED MAIN MEMORY PARITY ERROR THRU CACHVEC (#114)
                                                        ;WILL RETRY TEST ONCE
001366  033727                  DH3                     ;PARITY  ADDRESS   MAINTEN CONTROL
                                                        ;CONDITN REFERENCD REGISTR REGISTR TESTNO PC AT ABORT
001370  037006                  DT3                     ;PPARER,LOADDR,PMAINT,PCONTR,TESTNO,BADPC,0
001372  037615                  DF3                     ; 0, 2, 0, 0, 0, 0

                                ;ITEM 5
001374  025346                  EM5                     ;MEMORY MANAGEMENT TRAP, MEMORY MANAGEMENT STATUS REGISTERS
001376  034057                  DH5                     ;STATUS  AUTOI/D VIRTADR
                                                        ;REGISTR REGISTR REGISTR TESTNO  PC AT ABORT
001400  037036                  DT5                     ;PMMR0,PMMR1,PMMR2,TESTNO,BADPC,0
001402  037623                  DF5                     ; 0, 0, 0, 0, 0

                                ;ITEM 6
001404  025454                  EM6                     ;SUMMARY OF MAP REGISTERS THAT TIMED OUT ON READ
001406  034163                  DH6                     ;REGADRS REGADRS
                                                        ; "OR"     "AND"  #ERRORS TESTNO  ERRORPC
```

```
621  001410  037052          DT6                     ;ADDROR,ADRAND,ERRCNT,TFSTNO,$ERRPC,0
622  001412  037630          DF6                     ; 0, 0, 1, 0, 0
623
624                          ;ITEM 7
625  001414  025534          EM7                     ;SUMMARY OF CACHE REGISTERS THAT TIMED OUT ON READ
626  001416  034163          DH6                     ;REGADRS REGADRS
                                                     ; "OR"    "AND"  #ERRORS TESTNO  ERRORPC
627
628  001420  037052          DT6                     ;ADDROR,ADRAND,ERRCNT,TESTNO,$ERRPC,0
629  001422  037630          DF6                     ; 0, 0, 1, 0, 0
630
631                          ;ITEM 10
632  001424  025616          EM10                    ;SUMMARY OF MAP REGISTERS NOT HOLDING ZERO IN LOW 16 BITS
633  001426  034253          DH10                    ;REGADRS REGADRS RECEIVD RECEIVD
                                                     ; "OR"    "AND"  "OR"    "AND"  #ERRORS TESTNO  ERRORPC
634
635  001430  037066          DT10                    ;ADDROR,ADRAND,DATAOR,DATAND,ERRCNT,TESTNO,$ERRPC,0
636  001432  037635          DF10                    ; 0, 0, 0, 0, 1, 0, 0
637
638                          ;ITEM 11
639  001434  025707          EM11                    ;SUMMARY OF MAP REGISTERS NOT HOLDING ZERO IN UPPER 6 BITS
640  001436  034253          DH10                    ;REGADRS REGADRS RECEIVD RECEIVD
                                                     ; "OR"    "AND"  "OR"    "AND"  #ERRORS TESTNO  ERRORPC
641
642  001440  037066          DT10                    ;ADDROR,ADRAND,DATAOR,DATAND,ERRCNT,TESTNO,$ERRPC,0
643  001442  037635          DF10                    ; 0, 0, 0, 0, 1, 0, 0
644
645                          ;ITEM 12
646  001444  026001          EM12                    ;POSSIBLE ERROR IN MAP REGISTER DATA PATH (MAP REG 00)
647  001446  034403          DH12                    ;COUNT    COUNT
                                                     ;EXPECTD RECEIVD TESTNO  ERRORPC
648
649  001450  037106          DT12                    ;$REG2,$REG0,TESTNO,$ERRPC,0
650  001452  037644          DF12                    ; 0, 0, 0, 0
651
652                          ;ITEM 13
653  001454  026067          EM13                    ;NOW PROBABLE ERROR IN MAP REGISTER DATA PATH (MAP REG 20)
654  001456  034403          DH12                    ;COUNT    COUNT
                                                     ;EXPECTD RECEIVD TESTNO  ERRORPC
655
656  001460  037120          DT13                    ;$REG3,$REG1,TESTNO,$ERRPC,0
657  001462  037644          DF12                    ; 0, 0, 0, 0
658
659                          ;ITEM 14
660  001464  026161          EM14                    ;SUMMARY OF DUAL ADDRESSING ERRORS ON LOADING MAP REGISTERS
661  001466  034461          DH14                    ;REGLOAD REGLOAD REGDUAL REGDUAL
                                                     ; "OR"    "AND"  "OR"    "AND"  #ERRORS TESTNO
662
663  001470  037132          DT14                    ;ADDROR,ADRAND,DATAOR,DATAND,ERRCNT,TESTNO,0
664  001472  037650          DF14                    ; 0, 0, 0, 0, 1, 0
665
666                          ;ITEM 15
667  001474  026254          EM15                    ;SUMMARY OF COUNT PATTERN FAILURES IN LOWER 16 BITS OF MAP REGIS
668  001476  034600          DH15                    ;MAPREG  MAPREG  EXPECTD EXPECTD RECEIVD RECEIVD
                                                     ; "OR"    "AND"  "OR"    "AND"  "OR"    "AND"  #ERRORS TESTNO
669
670  001500  037150          DT15                    ;ADDROR,ADRAND,PATTOR,PATAND,DATAOR,DATAND,ERRCNT,TESTNO,0
671  001502  037656          DF15                    ; 0, 0, 0, 0, 0, 0, 1, 0
672
673                          ;ITEM 16
674  001504  026360          EM16                    ;SUMMARY OF COUNT PATTERN FAILURES IN UPPER 6 BITS OF MAP REGIST
675  001506  034600          DH15                    ;MAPREG  MAPREG  EXPECTD EXPECTD RECEIVD RECEIVD
                                                     ; "OR"    "AND"  "OR"    "AND"  "OR"    "AND"  #ERRORS TESTNO
676
```

```
                                DI15                    ;ADDROR,ADRAND,PATTOR,PATAND,DATAOR,DATAND,ERRCNT,TESTNO,0
                                DF15                    ; 0, 0, 0, 0, 0, 0, 1, 0

                                ;ITEM 17
                                EM17                    ;COULD NOT CLEAR CACHE CONTROL REGISTER
                                                        ;POSSIBLE ERROR IN CACHE REGISTER DATA PATH
                                DH17                    ;RECEIVD TESTNO  ERRORPC
                                DT17                    ;$REG0,TESTNO,$ERRPC,0
                                DF17                    ; 0, 0, 0

                                ;ITEM 20
                                EM20                    ;COULD NOT CLEAR CACHE MAINTENENCE REGISTER
                                                        ;POSSIBLE ERROR IN CACHE REGISTER DATA PATH
                                DH17                    ;RECEIVD TESTNO  ERRORPC
                                DT20                    ;$REG1,TESTNO,$ERRPC,0
                                DF17                    ; 0, 0, 0

                                ;ITEM 21
                                EM21                    ;COULD NOT READ 177740 FROM CACHE LO ADDRESS REG (LOADRS)
                                                        ;POSSIBLE ERROR IN CACHE REGISTER DATA PATH
                                DH17                    ;RECEIVD TESTNO  ERRORPC
                                DT17                    ;$REG0,TESTNO,$ERRPC,0
                                DF17                    ; 0, 0, 0

                                ;ITEM 22
                                EM22                    ;COULD NOT READ 000003 FROM CACHE HI ADDRESS REG (HIADRS)
                                                        ;POSSIBLE ERROR IN CACHE REGISTER DATA PATH
                                DH17                    ;RECEIVD TESTNO  ERRORPC
                                DT17                    ;$REG0,TESTNO,$ERRPC,0
                                DF17                    ; 0, 0, 0

                                ;ITEM 23
                                EM23                    ;SUMMARY OF COUNT PATTERN FAILURES IN CACHE CONTROL REGISTER
                                DH23                    ;EXPECTD EXPECTD RECEIVD RECEIVD
                                                        ;  'OR'    'AND'   'OR'    'AND'   #ERRORS TESTNO
                                DT23                    ;PATTOR,PATAND,DATAOR,DATAND,ERRCNT,TESTNO,0
                                DF23                    ; 0, 0, 0, 0, 1, 0

                                ;ITEM 24
                                EM24                    ;SUMMARY OF COUNT PATTERN FAILURES IN CACHE MAINTENENCE REGISTER
                                DH23                    ;EXPECTD EXPECTD RECEVID RECEIVD
                                                        ;  'OR'    'AND'   'OR'    'AND'   #ERRORS TESTNO
                                DT23                    ;PATTOR,PATAND,DATAOR,DATAND,ERRCNT,TESTNO,0
                                DF23                    ; 0, 0, 0, 0, 1, 0

                                ;ITEM 25
                                EM25                    ;REFERENCED MAP REGISTER 0 WITH ADDRESS ONE BIT
                                                        ;DIFFERENT THAN 770200
                                DH25                    ;ADDRUSED  BITDIFF  TESTNO  ERRORPC
                                DT25                    ;$REG4,$REG0,TESTNO,$ERRPC,0
                                DF25                    ;3, 4, 0, 0

                                ;ITEM 26
                                EM26                    ;REFERENCED CACHE LOW ADDRESS REGISTER WITH
                                                        ;ADDRESS ONE BIT DIFFERENT THAN 777740
                                DH25                    ;ADDRUSED  BITDIFF  TESTNO  ERRORPC
```

```
001610  037230          DT25            ;$REG4,$REG0,TESTNO,$ERRPC,0
001612  037677          DF25            ;3, 4, 0, 0

                        ;ITEM 27
001614  027544          EM26            ;REFERENCED CACHE LO ADDRESS REGISTER WITH
                                        ;ONE BIT DIFFERENT THAN 777740
001616  035170          DH27            ;ADDRUSED  TESTNO  ERRORPC
001620  037242          DT27            ;$REG1,TESTNO,$ERRPC
001622  037703          DF27            ;3, 0, 0

                        ;ITEM 30
001624  027665          EM30            ;CAN'T GET TO MAIN MEMORY FROM UNIBUS WITH THE MAP OFF
                                        ;SO I'LL JUMP TO THE SIZE JUMPER TEST FOR VERIFICATION
001626  035222          DH30            ;TESTNO  ERRORPC
001630  037252          DT30            ;TESTNO,$ERRPC,0
001632  037706          DF30            ;0, 0

                        ;ITEM 31
001634  030041          EM31            ;SUMMARY OF COUNT PATTERN FAILURES ON THE UNIBUS DATA  PATH
001636  035007          DH23            ;EXPECTD EXPECTD RECEIVD RECEIVD
                                        ;  'OR'    'AND'   'OR'   'AND'  #ERRORS TESTNO
001640  037212          DT23            ;PATTOR,PATAND,DATAOR,DATAND,ERRCTN,TESTNO,0
001642  037671          DF23            ; 0, 0, 0, 0, 1, 0

                        ;ITEM 32
001644  030133          EM32            ;UNIBUS MAP IS RELOCATING WHEN NOT ENABLED
001646  035222          DH30            ;TESTNO  ERRORPC
001650  037252          DT30            ;TESTNO,$ERRPC,0
001652  037706          DF30            ; 0, 0

                        ;ITEM 33
001654  030205          EM33            ;CANNOT USE ANY OF THE MAP REGISTERS OR PHYSICAL
                                        ;ADDRESS BIT14 IS STUCK LOW. MUST RESTART PROGRAM
                                        ;IF YOU DON'T LOOP ON THIS PROBLEM.
001656  035222          DH30            ;TESTNO  ERRORPC
001660  037252          DT30            ;TESTNO,$ERRPC,0
001662  037706          DF30            ; 0, 0

                        ;ITEM 34
001664  030362          EM34            ;THE NUMBER OF MAP REGISTERS REMOVED BY JUMPER SETTING
                                        ;DOES NOT AGREE WITH THE NUMBER FOUND TO BE MISSING.
001666  035242          DH34            ;REMOVED MISSING TESTNO  ERRORPC
001670  037260          DT34            ;ERRCNT,CNTR,TESTNO,$ERRPC,0
001672  037710          DF34            ; 0, 0, 0, 0

                        ;ITEM 35
001674  030533          EM35            ;THE SIZE JUMPERS ON THE UNIBUS MAP ARE NOT SET
                                        ;IN THEIR DEFAULT POSITION.  THIS WOULD ALLOW UNIBUS
                                        ;ADDRESSES 000000 TO 757776 TO REFERENCE MAIN MEMORY
                                        ;THEIR CURRENT SETTING ALLOWS ONLY:
001676  035302          DH35            ;LOWEST  HIGHEST TESTNO  ERRORPC
001700  037272          DT35            ;LOWEST,HIGEST,TESTNO,$ERRPC,0
001702  037714          DF35            ; 4, 4, 0, 0

                        ;ITEM 36
001704  031020          EM36            ;MAP REGISTER UNDER TEST DID NOT RESPOND IN DUAL MAPPING TES
```

```
  789  0017J6  035342          DH36            ;TESTNO  ERRORPC UNIBUS ADDRESS OF MAP REGISTER UNDER TEST
  790  001710  037304          DT36            ;TESTNO,$ERRPC,$REG0,0
  791  001712  037720          DF36            ; 0, 0, 3
  792
  793                          ;ITEM 37
  794  001714  031115          EM37            ;SUMMARY OF UNIBUS ADDRESS ERRORS, WITH MAP RELOCATION DISABLED
  795  001716  035007          DH23            ;EXPECTD EXPECTD RECEIVD RECEIVD
                                                ;  'OR'    'AND'  'OR'    'AND'  #ERRORS TESTNO
  796
  797  001720  037314          DT37            ;ADDROR,ADRAND,DATAOR,DATAND,ERRCNT,TESTNO
  798  001722  037723          DF37            ; 0, 0, 0, 0, 1, 0
  799
  800                          ;ITEM 40
  801  001724  031220          EM40            ;MAIN MEMORY TIME OUT OVER THE UNIBUS DID NOT OCCUR PROPERLY.
  802  001726  035434          DH40            ;CONDITN CONDITN
                                                ;EXPECTD RECEIVD TESTNO  ERRORPC
  803
  804  001730  037332          DT40            ;CPUEXP,PCPUER,TESTNO,$ERRPC,0
  805  001732  037731          DF40            ; 0, 0, 0, 0
  806
  807                          ;ITEM 41
  808  001734  031513          EM41            ;RELOCATION THROUGH THE MAP WAS NOT CORRECT, FULL ADD.
  809  001736  035514          DH41            ;CORRECT ADDRESS
                                                ;ADDRESS FETCHED TESTNO  ERRORPC
  810
  811  001740  037344          DT41            ;$REG2,$REG1,TESTNO,$ERRPC
  812  001742  037735          DF41            ; 0, 0, 0, 0
  813
  814                          ;ITEM 42
  815  001744  031375          EM42            ;RELOCATION THRU THE MAP WAS NOT CORRECT, CARRY PROPAGATION
  816  001746  035574          DH42            ;CORRECT   EXPECTD RECEIVD
                                                ;ADDRESS   DATA    FROM UB TESTNO  ERRORPC
  817
  818  001750  037356          DT42            ;$REG1,$REG3,$REG2,TESTNO,$ERRPC
  819  001752  037741          DF42            ; 3, 0, 0, 0, 0
  820
  821                          ;ITEM 43
  822  001754  031470          EM43            ;THE TOP OF MEMORY IS DIFFERENT THAN THE SIZE JUMPERS
  823  001756  035674          DH43            ;SIZEJUMP TOPFOUND TESTNO ERRORPC
  824  001760  037372          DT43            ;SIZELO,RSIZE,TESTNO,$ERRPC,0
  825  001762  037746          DF43            ; 0, 0, 0, 0
  826
  827                          ;ITEM 44
  828  001764  031555          EM44            ;PARITY REPORTING THRU THE MAP IS NOT CORRECT
  829  001766  035734          DH44            ;CONDITN CONDITN ADDRESS   MAINTEN CONTROL
                                                ;EXPECTD RECEIVD REFERENCD REGISTR REGISTR TESTNO  ERRORPC
  830
  831  001770  037404          DT44            ;$TMP4,PPARER,PLOADR,PMAINT,PCONTR,TESTNO,$ERRPC,0
  832  001772  037752          DF44            ; 0, 0, 2, 0, 0, 0, 0
  833
  834                          ;ITEM 45
  835  001774  031632          EM45            ;MAIN MEMORY TIME OUT OVER THE UNIBUS DID NOT OCCUR PROPERLY.
  836                                          ;TEST BEING RUN OVER UNIBUS
  837  001776  035434          DH40            ;CONDITN CONDITN
                                                ;EXPECTD RECEIVD TESTNO  ERRORPC
  838
  839  002000  037332          DT40            ;CPUEXP,PCPUER,TESTNO,$ERRPC,0
  840  002002  037731          DF40            ; 0, 0, 0, 0
  841
  842                          ;ITEM 46
  843  002004  031765          EM46            ;RELOCATION THROUGH THE MAP WAS NOT CORRECT, FULL ADD.
  844                                          ;TEST BEING RUN OVER UNIBUS
```

```
002006  035514      DH41            ;CORRECT ADDRESS
                                    ;ADDRESS FETCHED TESTNO  ERRORPC
002010  037344      DT41            ;$REG2,$REG1,TESTNO,$ERRPC
002012  037735      DF41            ; 0, 0, 0, 0

                    ;ITEM 47
002014  032107      EM47            ;RELOCATION THRU THE MAP WAS NOT CORRECT, CARRY PROPAGATION
                                    ;TEST BEING RUN OVER UNIBUS
002016  035574      DH42            ;CORRECT   EXPECTD RECEIVD
                                    ;ADDRESS   DATA    FROM UB TESTNO  ERRORPC
002020  037356      DT42            ;$REG1,$REG3,$REG2,TESTNO,$ERRPC
002022  037741      DF42            ; 3, 0, 0, 0, 0

                    ;ITEM 50
002024  032242      EM50            ;THE TOP OF MEMORY IS DIFFERENT THAN THE SIZE JUMPERS
                                    ;TEST BEING RUN OVER UNIBUS
002026  035674      DH43            ;SIZJUMP TOPFOUND TESTNO
002030  037372      DT43            ;SIZELO,RSIZE,TESTNO,0
002032  03774[illegible]      DF43            ; 0, 0, 0


                    ;ITEM 51
002034  032367      EM51            ;PARITY REPORTING THRU THE MAP IS NOT CORRECT
                                    ;TEST BEING RUN OVER UNIBUS
002036  035734      DH44            ;CONDITN CONDITN ADDRESS   MAINTEN CONTROL
                                    ;EXPECTD RECEIVD REFERENCD REGISTR REGISTR TESTNO  ERRORPC
002040  037404      DT44            ;$TMP4,PPARER,PLOADR,PMAINT,PCONTR,TESTNO,$ERRPC,0
002042  037752      DF44            ; 0, 0, 2, 0, 0, 0, 0

                    ;ITEM 52
002044  032504      EM52            ;SUMMARY OF DUAL MAPPING ERRORS
002046  035007      DH23            ;EXPECTD EXPECTD RECEIVD RECEIVD
                                    ; "OR"    "AND"   "OR"    "AND"  #ERRORS TESTNO
002050  037314      DT37            ;ADDROR,ADRAND,DATAOR,DATAND,$ERRPC,TESTNO,0
002052  037723      DF37            ; 0, 0, 0, 0, 1, 0

                    ;ITEM   53

002054  032543      EM53    ;BYP BIT IN UBMR COULD NOT BE CLEARED
002056  036100      DH53
002060  037424      DT53
002062  037706      DF30

                    ;ITEM   54

002064  032610      EM54    ;BYP BIT IN UBMR COULD NOT BE SET
002066  036100      DH53
002070  037424      DT53
002072  037706      DF30

                    ;ITEM   55
002074  032651      EM55            ;MK11 CSR COULD NOT BE ACCESSED
002076  036127      DH55
002100  037434      DT55
002102  037606      DF1
```

```
                                        ;ITEM   56
                                        
002104  032725                          EM56                    ;TEST DATA REF NOT A HIT
002106  036172                          DH56
002110  037446                          DT56
002112  037644                          DF12
                                        
                                        ;ITEM   57
                                        
002114  032763                          EM57                    ;TEST DATA REF NOT A MISS
002116  036237                          DH57                    ;CACHE BYPASS ON UNIBUS MAP DID NOT INVALIDATE CACHED DATA
002120  037460                          DT57
002122  037606                          DF1
                                        
002124                                  ER200:                  ;THIS IS THE STARTING POINT FOR ERROR MESSAGES
                                                                ;201 THRU 377.  THEY ARE USED FOR MULTIPLE
                                                                ;ERROR MESSAGES.
                                        
                                        ;ITEM 201
002124  033122                          EM201                   ;THE FOLLOWING REGISTERS TIMED OUT WHEN READ
002126  036303                          DH201                   ;REGADRS TESTNO ERRORPC
002130  037472                          DT201                   ;$REG0,TESTNO,$ERRPC,0
002132  037761                          DF201                   ; 0, 0, 0
                                        
                                        ;ITEM 202
002134  033176                          EM202                   ;THE FOLLOWING MAP REGISTERS WILL NOT CLEAR
002136  036333                          DH202                   ;REGADRS DATAREC TESTNO  ERRORPC.
002140  037502                          DT202                   ;$REG0,$TMP0,TESTNO,$ERRPC,0
002142  037764                          DF202                   ; 0, 0, 0, 0
                                        
                                        ;ITEM 203
002144  033251                          EM203                   ;THE FOLLOWING ARE DUAL ADDRESSING ERRORS IN THE UNIBUS MAP
002146  036373                          DH203                   ;MAPREG  MAPREG
                                                                ;TESTING DUALED  TESTNO  ERRORPC
002150  037514                          DT203                   ;$REG0,$REG1,$TESTNO,$ERRPC,0
002152  037770                          DF203                   ; 0, 0, 0, 0
                                        
                                        ;ITEM 204
002154  033344                          EM204                   ;THE COUNT PATTERN THRU THE MAP REGISTERS FAILED
002156  036452                          DH204                   ;REGADRS PATTERN EXPECTD RECEIVD TESTNO  ERRORPC
002160  037526                          DT204                   ;$REG0,$REG2,$REG4,$REG3,TESTNO,$ERRPC,0
002162  037774                          DF204                   ; 0, 0, 0, 0, 0, 0
                                        
                                        ;ITEM 205
002164  033424                          EM205                   ;UNIBUS DATA PATH COUNT PATTERN FAILURE
002166  036532                          DH205                   ;EXPECTD RECEIVD ADDRSLOAD TESTNO  ERRORPC
002170  037544                          DT205                   ;$REG1,$REG0,$REG2,TESTNO,$ERRPC,0
002172  040002                          DF205                   ;0, 0, 3, 0, 0
                                        
                                        ;ITEM 206
002174  033473                          EM206                   ;UNIBUS ADDRESSING ERRORS, MAP RELOCATION DISABLED
002176  036604                          DH206                   ;ADDRESS ADDRESS
                                                                ;EXPECTD RECEIVD TESTNO ERRORPC
002200  037560                          DT206                   ;$REG0,$REG3,TESTNO,$ERRPC,0
002202  040007                          DF206                   ; 0, 0, 0, 0
```

```
                                ;ITEM 207
002204  033555                  EM207                   ;COUNT PATTERN FAILURES IN CACHE REGISTERS
002206  036664                  DH207                   ;REGISTR EXPECTD RECEIVD
                                                        ;ADDRESS  DATA    DATA   TESTNO  ERRORPC
002210  037572                  D1207                   ;$REG0,$REG2,$REG3,TESTNO,$ERRPC,0
002212  040013                  DF207                   ; 0, 0, 0, 0, 0
                                ;;**************************************************************
                                .SBTTL  SCOPE HANDLER ROUTINE
                                ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
                                ;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
                                ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
                                ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
                                ;*SW14=1         LOOP ON TEST
                                ;*SW11=1         INHIBIT ITERATIONS
                                ;*SW09=1         LOOP ON ERROR
                                ;*SW08=1         LOOP ON TEST IN SWR<6:0>
                                ;*CALL
                                ;*      SCOPE           ;;SCOPE=IOT
002214                          $SCOPE:
002214  005037  001322                  CLR     RETRY           ;CLEAR RETRY FLAG AN THE START OF
                                                                ;EACH TEST
002220  005037  001254                  CLR     ERRCNT          ;CLEAR THE MULTIPLE ERROR COUNTER
002224  005037  001232                  CLR     DATAOR          ;LOCATION FOR LOGICAL OR OF BAD DATA
002230  005037  001226                  CLR     ADDROR          ;LOCATION FOR LOGICAL OR OF ADDRESS
002234  005037  001236                  CLR     PATTOR          ;LOCATION FOR LOGICAL OR OF PATTERN LOADED
002240  012700  177777                  MOV     #-1,R0          ;LOAD -1 INTO R0 TO INITIALIZE LOGICAL AND LOCS
002244  010037  001230                  MOV     R0,DATAND       ;LOCATION FOR LOGICAL AND OF BAD DATA
002250  010037  001224                  MOV     R0,ADRAND       ;LOCATION FOR LOGICAL AND OF ADDRESS
002254  010037  001234                  MOV     R0,PATAND       ;LOCATION FOR LOGICAL AND OF PATTERN LOADED
002260  006137  177570                  ROL     @#SWR           ;;LOOP ON PRESENT TEST?
002264  100514                          BMI     $OVER           ;;YES IF SW14=1
                                ;######START OF CODE FOR THE XOR TESTER######
002266  000416                  $XTSTR: BR      6$              ;;IF RUNNING ON THE 'XOR' TESTER CHANGE
                                                                ;;THIS INSTRUCTION TO A 'NOP' (NOP=240)
002270  013746  000004                  MOV     @#ERPVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
002274  012737  002314  000004          MOV     #5$,@#ERRVEC    ;;SET FOR TIMEOUT
002302  005737  177060                  TST     @#177060        ;;TIME OUT ON XOR?
002306  012637  000004                  MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
002312  000466                          BR      $SVLAD          ;;GO TO THE NEXT TEST
002314  022626                  5$:     CMP     (SP)+,(SP)+     ;;CLEAR THE STACK AFTER A TIME OUT
002316  012637  000004                  MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
002322  000426                          BR      7$              ;;LOOP ON THE PRESENT TEST
002324                          6$:;######END OF CODE FOR THE XOR TESTER######
002324  032737  000400  177570          BIT     #BIT08,@#SWR    ;;LOOP ON SPEC. TEST?
002332  001407                          BEQ     2$              ;;BR IF NO
002334  013746  177570                  MOV     @#SWR,-(SP)     ;;SET DESIRED TEST NUM. FROM SWR
002340  042716  000200                  BIC     #^C<SWRMSK>,(SP) ;;STRIP AWAY UNDESIRED BITS
002344  122637  001102                  CMPB    (SP)+,$TSTNM    ;;ON THE RIGHT TEST?
002350  001462                          BEQ     $OVER           ;;BR IF YES
002352  105737  001103          2$:     TSTB    $ERFLG          ;;HAS AN ERROR OCCURRED?
```

```
1013  002356  001421                    BEQ     3$                  ;;BR IF NO
1014  002360  123737  001115  001103    CMPB    $ERMAX,$ERFLG       ;;MAX. ERRORS FOR THIS TEST OCCURRED?
1015  002366  101015                    BHI     3$                  ;;BR IF NO
1016  002370  032737  001000  177570    BIT     #BIT09,@#SWR        ;;LOOP ON ERROR?
1017  002376  001404                    BEQ     4$                  ;;BR IF NO
1018  002400  013737  001110  001106 7$: MOV     $LPERR,$LPADR       ;;SET LOOP ADDRESS TO LAST SCOPE
1019  002406  000443                    BR      $OVER
1020  002410  105037  001103        4$: CLRB    $ERFLG              ;;ZERO THE ERROR FLAG
1021  002414  005037  001204            CLR     $TIMES              ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
1022  002420  000415                    BR      1$                  ;;ESCAPE TO THE NEXT TEST
1023  002422  032737  004000  177570 3$: BIT     #BIT11,@#SWR        ;;INHIBIT ITERATIONS?
1024  002430  001011                    BNE     1$                  ;;BR IF YES
1025  002432  005737  001100            TST     $PASS               ;;IF FIRST PASS OF PROGRAM
1026  002436  001406                    BEQ     1$                  ;;      INHIBIT ITERATIONS
1027  002440  005237  001104            INC     $ICNT               ;;INCREMENT ITERATION COUNT
1028  002444  023737  001204  001104    CMP     $TIMES,$ICNT        ;;CHECK THE NUMBER OF ITERATIONS MADE
1029  002452  002021                    BGE     $OVER               ;;BR IF MORE ITERATION REQUIRED
1030  002454  012737  000001  001104 1$: MOV     #1,$ICNT            ;;REINITIALIZE THE ITERATION COUNTER
1031  002462  013737  002532  001204    MOV     $MXCNT,$TIMES       ;;SET NUMBER OF ITERATIONS TO DO
1032  002470  105237  001102        $SVLAD: INCB    $TSTNM          ;;COUNT TEST NUMBERS
1033  002474  011637  001106                MOV     (SP),$LPADR     ;;SAVE SCOPE LOOP ADDRESS
1034  002500  011637  001110                MOV     (SP),$LPERR     ;;SAVE ERROR LOOP ADDRESS
1035  002504  005037  001206                CLR     $ESCAPE         ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
1036  002510  112737  000001  001115        MOVB    #1,$ERMAX       ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
1037  002516  013737  001102  177570 $OVER: MOV     $TSTNM,@#DISPLAY ;;DISPLAY TEST NUMBER
1038  002524  013716  001106                MOV     $LPADR,(SP)     ;;FUDGE RETURN ADDRESS
1039  002530  000002                        RTI                     ;;FIXES PS
1040  002532  000144                $MXCNT: 100.                    ;;MAX. NUMBER OF ITERATIONS
                                  ;;**************************************************************
1041
1042                              .SBTTL  ERROR HANDLER ROUTINE
1043
1044                              ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
1045                              ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
1046                              ;*AND GO TO ERTYPE ON ERROR
1047                              ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
1048                              ;*SW15=1        HALT ON ERROR
1049                              ;*              HALT CAN OCCUR BEFORE AND AFTER THE ERROR TYPEOUT
1050                              ;*SW13=1        INHIBIT ERROR TYPEOUTS
1051                              ;*SW10=1        BELL ON ERROR
1052                              ;*SW09=1        LOOP ON ERROR
1053                              ;*CALL
1054                              ;*      ERROR   N       ;;ERROR=EMT AND N=ERROR ITEM NUMBER
1055
1056
1057  002534                  $ERROR:
1058  002534  113737  001102  001262        MOVB    $TSTNM,TESTNO   ;SAVE TEST NUMBER FOR ERROR TYPE OUT
1059  002542  005237  001254                INC     ERRCNT          ;COUNT ALL MULTIPLE ERRORS
1060  002546  010037  001154                MOV     R0,$REG0        ;SAVE R0 FOR POSSIBLE TYPE OUT
1061  002552  010137  001156                MOV     R1,$REG1        ;SAVE R1 FOR POSSIBLE TYPE OUT
1062  002556  010237  001160                MOV     R2,$REG2        ;SAVE R2 FOR POSSIBLE TYPE OUT
1063  002562  010337  001162                MOV     R3,$REG3        ;SAVE R3 FOR POSSIBLE TYPE OUT
1064  002566  010437  001164                MOV     R4,$REG4        ;SAVE R4 FOR POSSIBLE TYPE OUT
1065  002572  010537  001166                MOV     R5,$REG5        ;SAVE R5 FOR POSSIBLE TYPE OUT
1066  002576  105237  001103        7$:     INCB    $ERFLG          ;;SET THE ERROR FLAG
1067  002602  001775                        BEQ     7$              ;;DON'T LET THE FLAG GO TO ZERO
1068  002604  013737  001102  177570        MOV     $TSTNM,@#DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
```

```
1069  002612  005737  177570            TST     @#SWR           ;;HALT ON ERROR   1?
1070  002616  100001                    BPL     8$              ;;BRANCH IF NO
1071  002620  000000                    HALT                    ;;YES--HALT
1072  002622  032737  002000  177570 8$: BIT     #BIT10,@#SWR    ;;BELL ON ERROR?
1073  002630  001402                    BEQ     1$              ;;NO - SKIP
1074  002632  104400  001210            TYPF    ,$BELL          ;;RING BELL
1075  002636  005237  001112         1$: INC     $ERTTL          ;;COUNT THE NUMBER OF ERRORS
1076  002642  011637  001116            MOV     (SP),$ERRPC     ;;GET ADDRESS OF ERROR INSTRUCTION
1077  002646  162737  000002  001116    SUB     #2,$ERRPC
1078  002654  117737  176236  001114    MOVB    @$ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
1079  002662  032737  020000  177570    BIT     #BIT13,@#SWR    ;;SKIP TYPEOUT IF SET
1080  002670  001004                    BNE     2$              ;;SKIP TYPEOUTS
1081  002672  004737  003036            JSR     PC,ERTYPE       ;;GO TO USER ERROR ROUTINE
1082  002676  104400  001215            TYPE    ,$CRLF
1083  002702  005737  177570         2$: TST     @#SWR           ;;HALT ON ERROR
1084  002706  100001                    BPL     9$              ;;SKIP IF CONTINUE
1085  002710  000000                    HALT                    ;;HALT ON ERROR!
1086  002712  022737  024600  000042 9$: CMP     #$ENDAD,42      ;;ACT-11?
1087  002720  001001                    BNE     3$              ;;BRANCH IF NO
1088  002722  000000                    HALT                    ;;YES
1089  002724  032737  001000  177570 3$: BIT     #BIT09,@#SWR    ;;LOOP ON ERROR SWITCH SET?
1090  002732  001402                    BEQ     4$              ;;BR IF NO
1091  002734  013716  001110            MOV     $LPERR,(SP)     ;;FUDGE RETURN FOR LOOPING
1092  002740  005737  001206         4$: TST     $ESCAPE         ;;CHECK FOR AN ESCAPE ADDRESS
1093  002744  001402                    BEQ     5$              ;;BR IF NONE
1094  002746  013716  001206            MOV     $ESCAPE,(SP)    ;;FUDGE RETURN ADDRESS FOR ESCAPE
1095  002752                         5$:
1096  002752  032737  001000  177570    BIT     #SW9,SWR        ;ARE YOU LOOPING ON THIS ERROR?
1097  002760  001425                    BEQ     EREXIT          ;BRANCH IF NOT LOOPING ON ERROR
1098  002762  012737  177777  177766    MOV     #-1,@#CPUERR    ;CLEAR CPU ERROR REGISTER
1099  002770  012737  177777  177744    MOV     #-1,@#MEMERR    ;CLEAR MEMORY ERROR REGISTER
1100  002776  042737  177776  177572    BIC     #177776,@#MMR0  ;CLEAR MEMORY MANAGEMENT STATUS REGISTER
1101  003004  012737  177777  005154    MOV     #-1,TOFLAG      ;INITIALIZE TRAP FLAG
1102  003012  012737  177777  005654    MOV     #-1,CPFLAG      ;INITIALIZE CP TRAP FLAG
1103  003020  012737  177777  005766    MOV     #-1,PAFLAG      ;INITIALIZE PARITY TRAP FLAG
1104  003026  012737  177777  006212    MOV     #-1,MMFLAG      ;INITIALIZE MEMORY MANAGEMENT TRAP FLAG
1105  003034  000002            EREXIT: RTI                     ;RETURN TO TEST
1106
1107
1108                                    .SBTTL  ERROR MESSAGE TYPE OUT ROUTINE
1109
1110
1111                                    ;*      THIS SUBROUTINE IS CALLED BY THE ERROR HANDLER TO TYPE
1112                                    ;*      THE ERROR MESSAGES.  IT PICKS UP THE ITEM BYTE ($ITEMB) NUMBER
1113                                    ;*      AND USES THAT TO INDEX THROUGH THE ERROR TABLE.  THE ERROR
1114                                    ;*      TABLE STARTS AT 'SERRTB' AND HAS FOUR (4) POINTERS FOR EACH
1115                                    ;*      ENTRY, 'EM', 'DH', 'DT', 'DF'.  THE 'EM' POINTS TO THE ERROR
1116                                    ;*      MESSAGE WHICH IS AN ASCIZ STRING.  THE 'DH' POINTS TO THE DATA
1117                                    ;*      HEADER WHICH IS ANOTHER ASCIZ STRING.  THE 'DT' POINTS TO THE
1118                                    ;*      DATA TABLE WHICH IS A GROUP OF WORDS CONTAINING THE ADDRESSES
1119                                    ;*      OF THE DATA TO BY TYPED.  THE FORMAT OF THIS DATA IS
1120                                    ;*      CONTROLLED BY THE 'DF' WHICH IS THE POINTER TO THE DATA FORMAT.
1121                                    ;*      THE DATA FORMAT IS A GROUP OF BYTES WHICH CONTAIN NUMBERS
1122                                    ;*      THAT CORRESPOND TO DIFFERENT TYPING FORMATS.
1123                                    ;*      0       -16 BIT OCTAL FORMAT
1124                                    ;*      1       -DECIMAL FORMAT
```

```
1125                                    ;*      2       -22 BIT OCTAL FORMAT.  DATA IS LOWER 16 BITS OF THE
1126                                    ;*               PHYSICAL ADDRESS, UPPER 6 BITS ARE ADJACENT TO LOWER 16
1127                                    ;*      3       -22 BIT OCTAL FORMAT.  DATA IS THE 16 BIT VIRTUAL
1128                                    ;*               ADDRESS IN KERNEL I-SPACE.
1129                                    ;*      4       -18 BIT OCTAL FORMAT.  DATA IS A 16 BIT NUMBER THAT
1130                                    ;*               WILL BE CONVERTED INTO A UNIBUS ADDRESS BY LEFT
1131                                    ;*               SHIFTING IT 6 BITS.
1132  003036  010046                   ERTYPE: MOV     R0,-(KSP)       ;SAVE R0 ON STACK
1133  003040  005000                           CLR     R0              ;CLEAR R0
1134  003042  113700  001114                  MOVB    @#$ITEMB,R0     ;PUT ITEM NUMBER IN R0
1135  003046  001004                          BNE     1$              ;BRANCH IF IT IS NON-ZERO
1136  003050  013746  001116                  MOV     $ERRPC,-(KSP)   ;PUT ERROR PC ON STACK FOR TYPING
1137  003054  104402                          TYPOC                   ;TYPE FAILING PC
1138  003056  000526                          BR      13$             ;GO TO RETURN
1139  003060  005300                   1$:    DEC     R0              ;ADJUST ITEM NUMBER TO BE A POINTER
1140  003062  072027  000003                  ASH     #3,R0           ;LEFT SHIFT ITEM NO. 3 PLACES
1141  003066  100024                          BPL     22$             ;BRANCH IF ITEM #LESS THAN 200
1142  003070  032700  001000                  BIT     #BIT9,R0        ;SEE IF ITEM # WAS 3XX
1143  003074  001415                          BEQ     21$             ;BRANCH IF ITEM # WAS 2XX
1144                                                                  ;AND TYPE ERROR MESSAGE
1145  003076  032737  000200  177570          BIT     #SW7,@#SWR      ;SEE IF SWITCH 7 IS UP
1146  003104  001404                          BEQ     20$             ;BRANCH IF SWITCH IS NOT UP
1147                                                                  ;AND TYPE DATA, ON MULTIPLE ERRORS
1148  003106  062766  000004  000002          ADD     #4,2(KSP)       ;SKIP 'TYPE  ,$CRLF' IF SW 7 IS UP
1149                                                                  ;INHIBIT MULTIPLE ERROR TYPEOUTS
1150  003114  000507                          BR      13$             ;BRANCH TO EXIT
1151  003116  042700  177000          20$:    BIC     #177000,R0      ;CLEAR UPPER BYTE OF R0
1152  003122  062700  002130                  ADD     #ER200+4,R0     ;POINT TO DATA TABLE ENTRY
1153  003126  000424                          BR      5$              ;GO TYPE DATA TABLE
1154  003130  042700  177000          21$:    BIC     #177000,R0      ;CLEAR UPPER BYTE OF R0
1155  003134  062700  000570                  ADD     #<ER200-$ERRTB>,R0 ;ADD DIFFERENCE BETWEEN
1156                                                                  ;ITEM 1 AND ITEM 201
1157                                    ;;                      GET POINTER TO ERROR MESSAGE AND TYPE IT
1158                                    ;;                      IF THE POINTER IS NOT ZERO
1159  003140  062700  001334          22$:    ADD     #$ERRTB,R0      ;ADD BASE OF ERROR TABLE
1160  003144  012037  003154                  MOV     (R0)+,2$        ;PUT MESSAGE POINTER IN TYPE STATEMENT
1161  003150  001404                          BEQ     3$              ;BRANCH IF NO ERROR MESSAGE
1162  003152  104400                          TYPE                    ;TYPE ERROR MESSAGE
1163  003154  000000                   2$:    .WORD   0               ;POINTER TO ERROR MESSAGE
1164  003156  104400  001215                  TYPE    ,$CRLF          ;TYPE CRLF
1165                                    ;;                      GET THE POINTER TO THE DATA HEADER AND
1166                                    ;;                      TYPE IT IF THE POINTER IS NOT ZERO
1167  003162  012037  003172          3$:     MOV     (R0)+,4$        ;PUT HEADER POINTER IN TYPE STATEMENT
1168  003166  001404                          BEQ     5$              ;BRANCH IF NO DATA HEADER
1169  003170  104400                          TYPE                    ;TYPE THE DATA HEADER
1170  003172  000000                   4$:    .WORD   0               ;POINTER TO DATA HEADER
1171  003174  104400  001215                  TYPE    ,$CRLF          ;TYPE CRLF
1172                                    ;;                      THIS IS THE START OF THE DATA OUTPUT IF THE
1173                                    ;;                      DATA POINTER IS NOT ZERO.  R0 POINTS TO THE
1174                                    ;;                      DATA FORMAT, R1 POINTS TO THE ADDRESS OF
1175                                    ;;                      THE DATA WORDS.
1176  003200  010146                   5$:    MOV     R1,-(KSP)       ;SAVE R1 ON THE STACK
1177  003202  012001                          MOV     (R0)+,R1        ;PUT DATA TABLE POINTER IN R1
1178  003204  001452                          BEQ     12$             ;BRANCH IF NO DATA TABLE
1179  003206  012000                          MOV     (R0)+,R0        ;PICK UP DATA FORMAT POINTER
1180  003210  105710                   6$:    TSTB    (R0)            ;IS THIS WORD OCTAL
```

```
003212  001003                  BNE     7$                      ;BRANCH IF NOT 16-BIT OCTAL
                        ;;              THIS IS 16 BIT OCTAL FORMAT (DF = 0)
003214  013146                  MOV     @(R1)+,-(KSP)           ;PUT WORD ON STACK FOR TYPING
003216  104402                  TYPOC                           ;TYPE THE WORD ON STACK AS 16 BIT OCTAL
003220  000436                  BR      11$                     ;GET READY FOR NEXT WORD
003222  122710  000001  7$:     CMPB    #1,(R0)                 ;IS THE WORD DECIMAL
003226  001003                  BNE     8$                      ;BRANCH IF NOT DECIMAL
                        ;;              THIS IS DECIMAL FORMAT (DF = 1)
003230  013146                  MOV     @(R1)+,-(KSP)           ;PUT WORD ON STACK FOR TYPING
003232  104410                  TYPDS                           ;TYPE THE WORD ON STACK AS DECIMAL
003234  000430                  BR      11$                     ;GET READY FOR NEXT WORD
003236  122710  000002  8$:     CMPB    #2,(R0)                 ;IS WORD 22-BIT PHYSICAL ADDRESS
003242  001012                  BNE     9$                      ;BRANCH IF NOT 22-BIT PHYSICAL ADDR
                        ;;              THIS IS 22-BIT PHYSICAL FORMAT (DF = 2)
003244  012146                  MOV     (R1)+,-(KSP)            ;PUT ADDR OF LOW WORD ON STACK
003246  004737  004750          JSR     PC,$DB2O                ;CONVERT NUMBER TO OCTAL ASCIZ
003252  062716  000003          ADD     #3,(KSP)                        ;ONLY WANT 8 DIGITS
003256  012637  003264          MOV     (KSP)+,30$              ;PUT POINTER AFTER 'TYPE' CALL
003262  104400                  TYPE                            ;TYPE ASCIZ STRING
003264  000000          30$:    .WORD   0                       ;WORD HOLDS POINTER TO ASCIZ STRING
003266  000413                  BR      11$                     ;GET READY FOR NEXT WORD
003270  122710  000003  9$:     CMPB    #3,(R0)                 ;IS THIS A 16-BIT VIRTUAL  ADDRESS
003274  001004                  BNE     10$                     ;BRANCH IF NOT 16-BIT VIRT. ADDR.
                        ;;              THIS IS 22-BIT VIRTUAL ADDRESS FORMAT
                        ;;              KERNEL I-SPACE ASSUMED.  (DF = 3)
003276  013146                  MOV     @(R1)+,-(KSP)           ;PUT 16-BIT VIRTUAL ADDR ON STACK
003300  004737  003344          JSR     PC,TYPVAD               ;GO TYPE 22-BIT ADDRESS FROM 16-BIT V.A.
003304  000404                  BR      11$                     ;GET READY FOR NEXT WORD
                        ;;              THIS IS FORMAT 4.  DATA WORD IS A UNIBUS ADDRESS
                        ;;              OUTPUT WILL BE 18-BITS WORD LEFT SHIFTED 6.
003306  013146          10$:    MOV     @(R1)+,-(KSP)           ;PUSH 16-BIT UNIBUS ADDRESS ON STACK
003310  004737  003452          JSR     PC,UBADDR               ;CONVERT TO 18-BIT UNIBUS ADDR AND TYPE
003314  000400                  BR      11$                     ;GET READY FOR NEXT WORD
003316  005200          11$:    INC     R0                      ;POINT TO NEXT FORMAT BYTE
003320  104400  003340          TYPE    ,32$                    ;TYPE TWO SPACES
003324  005711                  TST     (R1)                    ;IS THERE ANOTHER WORD?
003326  001401                  BEQ     12$                     ;BRANCH IF ALL DONE
003330  000727                  BR      6$                      ;GO BACK FOR NEXT NUMBER
003332  012601          12$:    MOV     (KSP)+,R1               ;RESTORE R1
003334  012600          13$:    MOV     (KSP)+,R0               ;RESTORE R0
003336  000207                  RTS     PC                      ;RETURN TO ERROR ROUTINE
003340  020040  000     32$:    .ASCIZ  ?  ?                    ;TWO SPACES
        003344                  .EVEN


                        .SBTTL  CONVERT 16-BIT VIRTUAL ADDRESS TO 22-BIT PHYSICAL ADDRESS
                        ;*
                        ;*      THIS ROUTINE IS CALLED BY A 'JSR PC' AFTER THE VIRTUAL ADDRESS
                        ;*      IS PUSHED ON THE KERNEL STACK.  THE V.A. IS THEN LOADED INTO
                        ;*      R1 AND THE UPPER 3 BITS ARE SHIFTED INTO R0 TO SELECT THE
                        ;*      CORRECT KERNEL I-SPACE PAR.  THE LOWER 12 BITS OF THE VIRTUAL
                        ;*      ADDRESS ARE ADDED TO THE PAR AS THEY ARE BY MEMORY MANAGEMENT
                        ;*      AND THE PHYSICAL ADDRESS IS SAVED IN MEMORY TO BE CONVERTED
                        ;*      TO ASCIZ AND TYPED.
                        ;*
003344  104412  TYPVAD: SAVREG                          ;SAVE ALL REGISTERS
```

```
1237  003346  016601  000002          MOV     2(KSP),R1       ;PUT VIRTUAL ADDR IN R1
1238  003352  005000                  CLR     R0              ;CLEAR R0 FOR CALCULATIONS
1239  003354  073027  000003          ASHC    #3,R0           ;LEFT SHIFT R0,R1 3 PLACES
1240  003360  006300                  ASL     R0              ;LEFT SHIFT R0 ONE MORE PLACE
1241  003362  006001                  ROR     R1              ;RIGHT SHIFT R1 SO OFFSET IS CORRECT
1242  003364  006001                  ROR     R1              ;RIGHT SHIF  R1
1243  003366  006001                  ROR     R1              ;RIGHT SHIFT R1
1244  003370  062700  172340          ADD     #KIPAR0,R0      ;FORM DESIRED PAR ADDR IN R0
1245  003374  011003                  MOV     (R0),R3         ;PUT CONTENTS OF PAR IN R3
1246  003376  005002                  CLR     R2              ;CLEAR R2 FOR PHYSICAL ADDR CALCULATIONS
1247  003400  073227  000006          ASHC    #6,R2           ;LEFT SHIFT <R2,R3> 6 PLACES
1248  003404  060103                  ADD     R1,R3           ;ADD OFFSET IN R1 TO BASE IN R3
1249  003406  005502                  ADC     R2              ;ADD ANY POSSIBLE CARRY TO UPPER 6 BITS
1250  003410  010237  001222          MOV     R2,PADRSH       ;PUT UPPER 6 BITS OF ADDR IN CORE
1251  003414  010337  001220          MOV     R3,PADRSL       ;PUT LOWER 16 BITS OF ADDR IN CORE
1252  003420  012746  001220          MOV     #PADRSL,-(KSP)  ;PUT POINTER TO LOWER 16 BITS ON STACK
1253  003424  004737  004750          JSR     PC,$DB2O        ;CONVERT NUMBER TO OCTAL ASCIZ
1254  003430  062716  000003          ADD     #3,(KSP)        ;ONLY TYPE 8 DIGITS
1255  003434  012637  003442          MOV     (KSP)+,3$       ;PUT POINTER AFTER TYPE INST
1256  003440  104400                  TYPE                    ;TYPE THE 22-BIT VIRTUAL ADDRESS
1257  003442  000000          3$:     .WORD   0               ;THIS WORD HOLDS THE POINTER TO
1258                                                          ;THE ASCIZ STRING
1259  003444  104414                  RESREG                  ;RESTORE ALL THE REGISTERS
1260  003446  012616                  MOV     (KSP)+,(KSP)    ;LEAVE ONLY RETURN ADDR ON STACK
1261  003450  000207                  RTS     PC              ;RETURN TO ERROR HANDLER
1262
1263
1264                          ;*THIS SUBROUTINE IS USED TO CONVERT THE A WORD PUSHED
1265                          ;*ON THE STACK INTO A UNIBUS ADDRESS AND TYPE IT AS A
1266                          ;*6 DIGIT NUMBER.  IT USES R1 & R0 AND LEAVES
1267                          ;*ALL OTHER REGISTERS UNCHANGED.
1268  003452  016601  000002  UBADDR: MOV     2(KSP),R1       ;LOAD 16 BIT ADDRESS INTO R1
1269  003456  005000                  CLR     R0              ;CLEAR R0 FOR CALCULATIONS
1270  003460  073027  000006          ASHC    #6,R0           ;LEFT SHIFT <R0:R1> 6 PLACES
1271  003464  010137  001220          MOV     R1,PADRSL       ;PUT LOWER 16 BITS IN PADRSL
1272  003470  010037  001222          MOV     R0,PADRSH       ;PUT UPPER 6 BITS IN PADRSH
1273  003474  012746  001220          MOV     #PADRSL,-(KSP)  ;PUSH POINTER TO WORDS ON STACK
1274  003500  004737  004750          JSR     PC,$DB2O        ;JUMP TO CONVERT ROUTINE
1275  003504  062716  000005          ADD     #5,(KSP)        ;ONLY USE LOWER 6 CHARS.
1276  003510  012637  003516          MOV     (KSP)+,3$       ;PUT POINTER AFTER TYPE CALL.
1277  003514  104400                  TYPE
1278  003516  000000          3$:     .WORD   0               ;HOLDS POINTER TO FIRST CHAR.
1279  003520  012616                  MOV     (KSP)+,(KSP)    ;LEAVE ONLY RETURN ADDRESS ON STACK
1280  003522  000207                  RTS     PC              ;RETURN TO ERROR TYPE ROUTINE.
1281
1282
1283                          ;;*****************************************************************
1284
1285                          .SBTTL  SAVE AND RESTORE R0-R5 ROUTINES
1286
1287                          ;*SAVE R0-R5
1288                          ;*CALL:
1289                          ;*      SAVREG
1290                          ;*UPON RETURN FROM $SAVREG THE STACK WILL LOOK LIKE:
1291                          ;*
1292                          ;*TOP---(+16)
```

```
1293                                    ;* +2---(+18)
1294                                    ;* +4---R5
1295                                    ;* +6---R4
1296                                    ;* +8---R3
1297                                    ;*+10---R2
1298                                    ;*+12---R1
1299                                    ;*+14---R0
1300
1301   003524                           $SAVREG:
1302   003524  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
1303   003526  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
1304   003530  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
1305   003532  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
1306   003534  010446                          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
1307   003536  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
1308   003540  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PS OF MAIN FLOW
1309   003544  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PC OF MAIN FLOW
1310   003550  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PS OF CALL
1311   003554  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PC OF CALL
1312   003560  000002                          RTI
1313
1314                                    ;*RESTORE R0-R5
1315                                    ;*CALL:
1316                                    ;*      RESREG
1317   003562                           $RESREG:
1318   003562  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PC OF CALL
1319   003566  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PS OF CALL
1320   003572  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PC OF MAIN FLOW
1321   003576  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PS OF MAIN FLOW
1322   003602  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
1323   003604  012604                          MOV     (SP)+,R4        ;;POP STACK INTO R4
1324   003606  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
1325   003610  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
1326   003612  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
1327   003614  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
1328   003616  000002                          RTI
1329                                    ;;******************************************************************
1330
1331                                    .SBTTL  TYPE ROUTINE
1332
1333                                    ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
1334                                    ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
1335                                    ;*NOTE1:        $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
1336                                    ;*NOTE2:        $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
1337                                    ;*NOTE3:        $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
1338                                    ;*
1339                                    ;*CALL:
1340                                    ;*1) USING A TRAP INSTRUCTION
1341                                    ;*      TYPE    ,MESADR         ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
1342                                    ;*OR
1343                                    ;*      TYPE
1344                                    ;*      MESADR
1345                                    ;*
1346                                    ;*2) USING A JSR INSTRUCTION
1347                                    ;*      MOV     PS,-(SP)        ;;PUSH PROCESSOR STATUS WORD ON THE STACK
1348                                    ;*      JSR     PC,$TYPE        ;;CALL TYPE ROUTINE
```

```
1349                                    ;*      MESADDR                 ;;FIRST ADRESS OF MESSAGE
1350
1351  003620  105737  001151            $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
1352  003624  100002                            BPL     1$              ;;BR IF YES
1353  003626  000000                            HALT                    ;;HALT HERE IF NO TERMINAL
1354  003630  000407                            BR      3$              ;;LEAVE
1355  003632  010046                    1$:     MOV     R0,-(SP)        ;;SAVE R0
1356  003634  017600  000002                    MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
1357  003640  112046                    2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
1358  003642  001005                            BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
1359  003644  005726                            TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
1360  003646  012600                            MOV     (SP)+,R0        ;;RESTORE R0
1361  003650  062716  000002            3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
1362  003654  000002                            RTI                     ;;RETURN
1363  003656  122716  000011            4$:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
1364  003662  001426                            BEQ     8$
1365  003664  122716  000200                    CMPB    #CRLF,(SP)      ;;BRANCH IF NOT
1366  003670  001004                            BNE     5$
1367  003672  005726                            TST     (SP)+           ;;POP  <CR><LF> EQUIV
1368  003674  104400  001215                    TYPE    ,$CRLF
1369  003700  000757                            BR      2$              ;;GET NEXT CHARACTER
1370  003702  004737  003764            5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
1371  003706  123726  001150            6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
1372  003712  001352                            BNE     2$              ;;IF NO GO GET NEXT CHAR.
1373  003714  013746  001146                    MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
1374                                                                    ;;AND THE NULL CHAR.
1375  003720  105366  000001            7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
1376  003724  002770                            BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
1377  003726  004737  003764                    JSR     PC,$TYPEC       ;;GO TYPE A NULL
1378  003732  105337  004030                    DECB    $CHARCNT        ;;DON'T COUNT THE NULL AS A CHARACTER
1379  003736  000770                            BR      7$              ;;LOOP
1380
1381                                    ;;HORIZONTAL TAB PROCESSOR
1382
1383  003740  112716  000040            8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
1384  003744  004737  003764            9$:     JSR     PC,$TYPEC       ;;TYPE A SPACE
1385  003750  132737  000007  004030            BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
1386  003756  001372                            BNE     9$              ;;TAB STOP
1387  003760  005726                            TST     (SP)+           ;;POP SPACE OFF STACK
1388  003762  000726                            BR      2$              ;;GET NEXT CHARACTER
1389  003764  105777  175152            $TYPEC: TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
1390  003770  100375                            BPL     $TYPEC
1391  003772  116677  000002  175144            MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
1392  004000  122766  000015  000002            CMPB    #CR,2(SP)       ;;BRANCH IF
1393  004006  001003                            BNE     1$              ;;NOT <CR>
1394  004010  105037  004030                    CLRB    $CHARCNT        ;;
1395  004014  000406                            BR      $TYPEX          ;;EXIT
1396  004016  122766  000012  000002    1$:     CMPB    #LF,2(SP)       ;;BRANCH IF
1397  004024  001402                            BEQ     $TYPEX          ;;<LF>
1398  004026  105227                            INCB    (PC)+           ;;INC SPACE
1399  004030  000000            $CHARCNT:.WORD  0                       ;;COUNT
1400  004032  000207            $TYPEX: RTS     PC
1401
1402                                    ;;******************************************************************
1403
1404                                    .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
```

```
                                        ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
                                        ;*OCTAL (ASCII) NUMBER AND TYPE IT.
                                        ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
                                        ;*CALL:
                                        ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                                        ;*      TYPOS                   ;;CALL FOR TYPEOUT
                                        ;*      .BYTE   N               ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
                                        ;*      .BYTE   M               ;;M=1 OR 0
                                        ;*                                      ;;1=TYPE LEADING ZEROS
                                        ;*                                      ;;0=SUPPRESS LEADING ZEROS
                                        ;*
                                        ;*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
                                        ;*$TYPOS OR $TYPOC
                                        ;*CALL:
                                        ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                                        ;*      TYPON                   ;;CALL FOR TYPEOUT
                                        ;*
                                        ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
                                        ;*CALL:
                                        ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                                        ;*      TYPOC                   ;;CALL FOR TYPEOUT

004034  017646  000000          $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
004040  116637  000001  004257          MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
004046  112637  004261                  MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
004052  062716  000002                  ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
004056  000406                          BR      $TYPON
004060  112737  000001  004257  $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
004066  112737  000006  004261          MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
004074  112737  000005  004256  $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
004102  010346                          MOV     R3,-(SP)        ;;SAVE R3
004104  010446                          MOV     R4,-(SP)        ;;SAVE R4
004106  010546                          MOV     R5,-(SP)        ;;SAVE R5
004110  113704  004261                  MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
004114  005404                          NEG     R4
004116  062704  000006                  ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
004122  110437  004260                  MOVB    R4,$OMODE       ;;SAVE IT FOR USE
004126  113704  004257                  MOVB    $OFILL R4       ;;GET THE ZERO FILL SWITCH
004132  016605  000012                  MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
004136  005003                          CLR     R3              ;;CLEAR THE OUTPUT WORD
004140  006105                  1$:     ROL     R5              ;;ROTATE MSB INTO 'C'
004142  000404                          BR      3$              ;;GO DO MSB
004144  006105                  2$:     ROL     R5              ;;FORM THIS DIGIT
004146  006105                          ROL     R5
004150  006105                          ROL     R5
004152  010503                          MOV     R5,R3
004154  006103                  3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
004156  105337  004260                  DECB    $OMODE          ;;TYPE THIS DIGIT?
004162  100016                          BPL     7$              ;;BR IF NO
004164  042703  177770                  BIC     #177770,R3      ;;GET RID OF JUNK
004170  001002                          BNE     4$              ;;TEST FOR 0
004172  005704                          TST     R4              ;;SUPPRESS THIS 0?
004174  001403                          BEQ     5$              ;;BR IF YES
004176  005204                  4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
004200  052703  000060                  BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
```

```
1461  004204  052703  000040          5$:     BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
1462  004210  110337  004254                  MOVB    R3,8$           ;;SAVE FOR TYPING
1463  004214  104400  004254                  TYPE    ,8$             ;;GO TYPE THIS DIGIT
1464  004220  105337  004256          7$:     DECB    $OCNT           ;;COUNT BY 1
1465  004224  003347                          BGT     2$              ;;BR IF MORE TO DO
1466  004226  002402                          BLT     6$              ;;BR IF DONE
1467  004230  005204                          INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
1468  004232  000744                          BR      2$              ;;GO DO THE LAST DIGIT
1469  004234  012605                  6$:     MOV     (SP)+,R5        ;;RESTORE R5
1470  004236  012604                          MOV     (SP)+,R4        ;;RESTORE R4
1471  004240  012603                          MOV     (SP)+,R3        ;;RESTORE R3
1472  004242  016666  000002  000004         MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
1473  004250  012616                          MOV     (SP)+,(SP)
1474  004252  000002                          RTI                     ;;RETURN
1475  004254     000          8$:     .BYTE   0               ;;STORAGE FOR ASCII DIGIT
1476  004255     000                  .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
1477  004256     000          $OCNT:  .BYTE   0               ;;OCTAL DIGIT COUNTER
1478  004257     000          $OFILL: .BYTE   0               ;;ZERO FILL SWITCH
1479  004260  000000          $OMODE: .WORD   0               ;;NUMBER OF DIGITS TO TYPE
1480                                  ;;**************************************************************
1481
1482                                  .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
1483
1484                                  ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
1485                                  ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
1486                                  ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
1487                                  ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
1488                                  ;*REPLACED WITH SPACES.
1489                                  ;*CALL:
1490                                  ;*      MOV     NUM,-(SP)       ;;PUT THE BINARY NUMBER ON THE STACK
1491                                  ;*      TYPDS                   ;;GO TO THE ROUTINE
1492
1493  004262                  $TYPDS:
1494  004262  010046                  MOV     R0,-(SP)        ;;PUSH R0 ON STACK
1495  004264  010146                  MOV     R1,-(SP)        ;;PUSH R1 ON STACK
1496  004266  010246                  MOV     R2,-(SP)        ;;PUSH R2 ON STACK
1497  004270  010346                  MOV     R3,-(SP)        ;;PUSH R3 ON STACK
1498  004272  010546                  MOV     R5,-(SP)        ;;PUSH R5 ON STACK
1499  004274  012746  020200          MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
1500  004300  016605  000020          MOV     20(SP),R5       ;;GET THE INPUT NUMBER
1501  004304  100004                  BPL     1$              ;;BR IF INPUT IS POS.
1502  004306  005405                  NEG     R5              ;;MAKE THE BINARY NUMBER POS.
1503  004310  112766  000055  000001  MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
1504  004316  005000          1$:     CLR     R0              ;;ZERO THE CONSTANTS INDEX
1505  004320  012703  004476          MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
1506  004324  112723  000040          MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
1507  004330  005002          2$:     CLR     R2              ;;CLEAR THE BCD NUMBER
1508  004332  016001  004466          MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
1509  004336  160105          3$:     SUB     R1,R5           ;;FORM THIS BCD DIGIT
1510  004340  002402                  BLT     4$              ;;BR IF DONE
1511  004342  005202                  INC     R2              ;;INCREASE THE BCD DIGIT BY 1
1512  004344  000774                  BR      3$
1513  004346  060105          4$:     ADD     R1,R5           ;;ADD BACK THE CONSTANT
1514  004350  005702                  TST     R2              ;;CHECK IF BCD DIGIT=0
1515  004352  001002                  BNE     5$              ;;FALL THROUGH IF 0
1516  004354  105716                  TSTB    (SP)            ;;STILL DOING LEADING 0'S?
```

```
1517  004356  100407                  BMI     7$              ;;BR IF YES
1518  004360  106316          5$:     ASLB    (SP)            ;;MSD?
1519  004362  103003                  BCC     6$              ;;BR IF NO
1520  004364  116663  000001  177777  MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
1521  004372  052702  000060          6$:     BIS     #'0,R2  ;;MAKE THE BCD DIGIT ASCII
1522  004376  052702  000040          7$:     BIS     #' ,R2  ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
1523  004402  110223                  MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
1524  004404  005720                  TST     (R0)+           ;;JUST INCREMENTING
1525  004406  020027  000010          CMP     R0,#10          ;;CHECK THE TABLE INDEX
1526  004412  002746                  BLT     2$              ;;GO DO THE NEXT DIGIT
1527  004414  003002                  BGT     8$              ;;GO TO EXIT
1528  004416  010502                  MOV     R5,R2           ;;GET THE LSD
1529  004420  000764                  BR      6$              ;;GO CHANGE TO ASCII
1530  004422  105726          8$:     TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
1531  004424  100003                  BPL     9$              ;;BR IF NO
1532  004426  116663  177777  177776  MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
1533  004434  105013          9$:     CLRB    (R3)            ;;SET THE TERMINATOR
1534  004436  012605                  MOV     (SP)+,R5        ;;POP STACK INTO R5
1535  004440  012603                  MOV     (SP)+,R3        ;;POP STACK INTO R3
1536  004442  012602                  MOV     (SP)+,R2        ;;POP STACK INTO R2
1537  004444  012601                  MOV     (SP)+,R1        ;;POP STACK INTO R1
1538  004446  012600                  MOV     (SP)+,R0        ;;POP STACK INTO R0
1539  004450  104400  004476          TYPE    ,$DBLK          ;;NOW TYPE THE NUMBER
1540  004454  016666  000002  000004  MOV     2(SP),4(SP)     ;;ADJUST THE STACK
1541  004462  012616                  MOV     (SP)+,(SP)
1542  004464  000002                  RTI                     ;;RETURN TO USER
1543  004466  023420          $DTBL:  10000.
1544  004470  001750                  1000.
1545  004472  000144                  100.
1546  004474  000012                  10.
1547  004476  000004          $DBLK:  .BLKW   4
1548                          ;;*****************************************************************
1549
1550                          .SBTTL  TRAP DECODER
1551
1552                          ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE 'TRAP' INSTRUCTION
1553                          ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
1554                          ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
1555                          ;*GO TO THAT ROUTINE.
1556
1557  004506  010046          $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
1558  004510  016600  000002          MOV     2(SP),R0        ;;GET TRAP ADDRESS
1559  004514  005740                  TST     -(R0)           ;;BACKUP BY 2
1560  004516  111000                  MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
1561  004520  016000  004526          MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
1562  004524  000200                  RTS     R0              ;;GO TO ROUTINE
1563
1564
1565                          .SBTTL  TRAP TABLE
1566
1567                          ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
1568                          ;*BY THE 'TRAP' INSTRUCTION.
1569
1570                          ;       ROUTINE
1571                          ;       -------
1572  004526          $TRPAD:
```

```
                        $TYPE   ;;CALL=TYPE      TRAP+0(104400)  TTY TYPEOUT ROUTINE
                        $TYPOC  ;;CALL=TYPOC     TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
                        $TYPOS  ;;CALL=TYPOS     TRAP+4(104404)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
                        $TYPON  ;;CALL=TYPON     TRAP+6(104406)  TYPE OCTAL NUMBER (AS PER LAST CALL)
                        $TYPDS  ;;CALL=TYPDS     TRAP+10(104410) TYPE DECIMAL NUMBER (WITH SIGN)
                        $SAVREG ;;CALL=SAVREG    TRAP+12(104412) SAVE R0-R5 ROUTINE
                        $RESREG ;;CALL=RESREG    TRAP+14(104414) RESTORE R0-R5 ROUTINE
                        TBITOFF ;;CALL=TBITO     TRAP+16(104416) THIS WILL TURN OFF T BIT TRAPPING
                        TBITRESTORE     ;;CALL=TBITR    TRAP+20(104420) THIS WILL RETURN THE T BIT TO PR

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

                .SBTTL  POWER DOWN AND UP ROUTINES

                ;POWER DOWN ROUTINE
004550 012737 004676 000024 $PWRDN: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
004556 012737 000340 000026         MOV     #340,@#PWRVEC+2 ;;PRIO:7
004564 010046                       MOV     R0,-(SP)        ;;PUSH R0 ON STACK
004566 010146                       MOV     R1,-(SP)        ;;PUSH R1 ON STACK
004570 010246                       MOV     R2,-(SP)        ;;PUSH R2 ON STACK
004572 010346                       MOV     R3,-(SP)        ;;PUSH R3 ON STACK
004574 010446                       MOV     R4,-(SP)        ;;PUSH R4 ON STACK
004576 010546                       MOV     R5,-(SP)        ;;PUSH R5 ON STACK
004600 010637 004702                MOV     SP,$SAVR6       ;;SAVE SP
004604 012737 004616 000024         MOV     #$PWRUP,@#PWRVEC ;;SET UP VECTOR
004612 000000                       HALT
004614 000776                       BR      .-2             ;;HANG UP

                ;POWER UP ROUTINE
004616 013706 004702 $PWRUP: MOV     $SAVR6,SP       ;;GET SP
004622 005037 004702         CLR     $SAVR6          ;;WAIT LOOP FOR THE TTY
004626 005237 004702 1$:     INC     $SAVR6          ;;WAIT FOR THE INC
004632 001375                BNE     1$              ;;OF  WORD
004634 012605                MOV     (SP)+,R5        ;;POP STACK INTO R5
004636 012604                MOV     (SP)+,R4        ;;POP STACK INTO R4
004640 012603                MOV     (SP)+,R3        ;;POP STACK INTO R3
004642 012602                MOV     (SP)+,R2        ;;POP STACK INTO R2
004644 012601                MOV     (SP)+,R1        ;;POP STACK INTO R1
004646 012600                MOV     (SP)+,R0        ;;POP STACK INTO R0
004650 012737 004550 000024  MOV     #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
004656 012737 000340 000026  MOV     #340,@#PWRVEC+2 ;;PRIO:7
004664 104400                TYPE                    ;;REPORT THE POWER FAILURE
004666 004704 $PWRMG: .WORD  PWRMSG                  ;;POWER FAIL MESSAGE POINTER
004670 012716                MOV     (PC)+,(SP)      ;;RESTART AT START
004672 010000 $PWRAD: .WORD  START                   ;;RESTART ADDRESS
004674 000002                RTI
004676 000000 $ILLUP: HALT                           ;;THE POWER UP SEQUENCE WAS STARTED
004700 000776                BR      .-2             ;; BEFORE THE POWER DOWN WAS COMPLETE
004702 000000 $SAVR6: 0                              ;;PUT THE SP HERE
004704 006412 047520 042527 PWRMSG: .ASCIZ  <12><15>?POWER FAILURE, RESTARTING PROGRAM?
004712 020122 040506 046111
004720 051125 026105 051040
004726 051505 040524 052122
004734 047111 020107 051120
004742 043517 040522 000115

                        .EVEN
```

```
1629
1630                                    ;;*****************************************************************
1631
1632                                    .SBTTL  DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE
1633
1634                                    ;*THIS ROUTINE WILL CONVERT A 32-BIT UNSIGNED BINARY NUMBER TO AN
1635                                    ;*UNSIGNED OCTAL ASCIZ NUMBER.
1636                                    ;*CALL
1637                                    ;*      MOV     #PNTR,-(SP)     ;;POINTER TO LOW WORD OF BINARY NUMBER
1638                                    ;*      JSR     PC,@#$DB20      ;;CALL THE ROUTINE
1639                                    ;*      RETURN                  ;;THE ADDRESS OF THE FIRST ASCIZ CHAR. IS ON THE STACK
1640
1641
1642  004750  104412                    $DB20:  SAVREG                  ;;SAVE ALL REGISTERS
1643  004752  016601  000002                    MOV     2(SP),R1        ;;PICKUP THE POINTER TO LOW WORD
1644  004756  012705  005067                    MOV     #$OCTVL+13.,R5  ;;POINTER TO DATA TABLE
1645  004762  012704  000014                    MOV     #12.,R4         ;;DO ELEVEN CHARACTERS
1646  004766  012703  177770                    MOV     #^C7,R3         ;;MASK
1647  004772  012100                            MOV     (R1)+,R0        ;;LOWER WORD
1648  004774  012101                            MOV     (R1)+,R1        ;;HIGH WORD
1649  004776  005002                            CLR     R2              ;;TERMINATOR
1650  005000  110245                    1$:     MOVB    R2,-(R5)        ;;PUT CHARACTER IN DATA TABLE
1651  005002  010002                            MOV     R0,R2           ;;GET THIS DIGIT
1652  005004  005304                            DEC     R4              ;;COUNT THIS CHARACTER
1653  005006  003007                            BGT     3$              ;;BR IF NOT THE LAST DIGIT
1654  005010  001405                            BEQ     2$              ;;BR IF IT IS THE LAST DIGIT
1655  005012  005205                            INC     R5              ;;ALL DIGITS DONE-ADJUST POINTER FOR FIRST
1656  005014  010566  000002                    MOV     R5,2(SP)        ;;ASCIZ CHAR. & PUT IT ON THE STACK
1657  005020  104414                            RESREG                  ;;RESTORE ALL REGISTERS
1658  005022  000207                            RTS     PC              ;;RETURN TO USER
1659  005024  006203                    2$:     ASR     R3              ;;POSITION THE MASK FOR THE LAST DIGIT
1660  005026  006001                    3$:     ROR     R1              ;;POSITION THE BINARY NUMBER FOR
1661  005030  006000                            ROR     R0              ;;      THE NEXT OCTAL DIGIT
1662  005032  006001                            ROR     R1
1663  005034  006000                            ROR     R0
1664  005036  006001                            ROR     R1
1665  005040  006000                            ROR     R0
1666  005042  040302                            BIC     R3,R2           ;;MASK OUT ALL JUNK
1667  005044  062702  000060                    ADD     #'0,R2          ;;MAKE THIS CHAR. ASCII
1668  005050  000753                            BR      1$              ;;GO PUT IT IN THE DATA TABLE
1669  005052  000016                    $OCTVL: .BLKB   14.             ;;RESERVE DATA TABLE
1670
1671
1672                                    .SBTTL  ************** SUBROUTINES UNIQUE TO THIS PROGRAM **************
1673
1674
1675
1676                                    .SBTTL  TURN OFF AND SAVE T-BIT
1677                                    ;;*****************************************************************
1678                                    ;*
1679                                    ;*      THIS SUBROUTINE IS REACHED BY THE TRAP CALL 'TBITO'. IT IS
1680                                    ;*      USED TO TURN OFF THE T-BIT IF IT IS ON.  THE PROCESSOR STATUS
1681                                    ;*      IS SAVED IN 'OLDPSW' SO THAT THE T-BIT CAN BE RESTORED TO ITS
1682                                    ;*      PREVIOUS STATUS WHEN CONDITIONS WARRANT.
1683                                    ;*
1684                                    ;;*****************************************************************
```

```
1685
1686  005070                                  TBITOFF:
 687  005070  032766  000020  000002                  BIT     #TBIT,2(KSP)    ;IS THE T-BIT ON?
1688  005076  001406                                  BEQ     1$              ;BRANCH TO EXIT IF IT IS NOT ON
1689  005100  016637  000002  001310                  MOV     2(KSP),OLDPSW   ;SAVE OLD PSW FOR RESTORING T BIT
1690  005106  042766  000020  000002                  BIC     #TBIT,2(KSP)    ;CLEAR T BIT IF IT IS ON
1691  005114  000006                          1$:     RTT                     ;RETURN TO PROGRAM
1692
1693
1694
1695                                          .SBTTL  RESTORE T-BIT TO ITS PREVIOUS CONDITION
1696                                          ;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1697                                          ;*
1698                                          ;*      THIS SUBROUTINE CAN BE REACHED BY THE TRAP CALL 'TBITR', IT IS
1699                                          ;*      USED TO RESTORE THE T-BIT AFTER A PARTICULAR TEST THAT CANNOT
1700                                          ;*      BE RUN WITH THE T-BIT ON.  IT USES THE PROCESSOR STATUS STORED
1701                                          ;*      IN 'OLDPSW' BY 'TBITO', REPLACES THE PS ON THE STACK WITH IT
1702                                          ;*      AND DOES AN 'RTT'.
1703                                          ;*
1704                                          ;---------------------------------------------------------------------------------------
1705  005116                                  TBITRESTORE:
1706  005116  013766  001310  000002                  MOV     OLDPSW,2(KSP)   ;PUT OLD PSW ON STACK
1707  005124  042737  000020  001310                  BIC     #TBIT,OLDPSW    ;CLEAR T-BIT IN 'OLDPSW' SO THAT
1708                                                                          ;IT WON'T BE TURNED ON BY ACCIDENT
1709  005132  000006                                  RTT                     ;RETURN TO PROGRAM AND INHIBIT
1710                                                                          ;T BIT TRAP AFTER THIS INSTRUCTION.
1711
1712
1713
1714                                          .SBTTL  SUBROUTINE TO CLEAR ALL OF THE MAP REGISTERS
1715                                          ;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1716                                          ;*
1717                                          ;*      THIS SUBROUTINE CLEARS ALL OF THE MAP REGISTERS BY LOADING
1718                                          ;*      THE ADDRESS OF MAPLOO INTO R3 AND THEN CLEARING THE
1719                                          ;*      REGISTER POINTED TO BY R3 UNTIL R3 POINTS ABOVE MAPH37.
1720                                          ;*
172;                                         ;---------------------------------------------------------------------------------------
1722  005134  012703  170200                  CLRMAP: MOV     #MAPLO,R3       ;PUT FIRST MAP ADDR IN R3
1723  005140  005023                          1$:     CLR     (R3)+           ;CLEAR MAP REGS
1724  005142  022703  170376                          CMP     #MAPH37,R3      ;SEE IF LAST ADDR IS IN R3
1725  005146  103374                                  BHIS    1$              ;BRANCH IF R3 LE THAN LAST ADDR
1726  005150  000207                                  RTS     PC              ;RETURN TO MAIN PROGRAM
1727
1728                                          .SBTTL  SUBROUTINE TO LOG AND REPORT TIMEOUTS OF MAP REGISTERS
1729                                          ;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1730                                          ;*
1731                                          ;*      THIS SUBROUTINE IS USED TO LOG AND REPORT THE FACT THAT A
1732                                          ;*      REFERENCE TO A MAPPING REGISTER TIMED OUT ON THE UNIBUS.  IT
1733                                          ;*      KEEPS A 'LOGICAL AND'' AND A 'LOGICAL OR'' OF EACH ADDRESS THAT
1734                                          ;*      TIMES OUT.
1735                                          ;*
1736                                          ;---------------------------------------------------------------------------------------
1737  005152                                  TIMEOUT:                        ;STARTING ADDRESS OF SUBROUTINE
1738  005152  005227                                  INC     (PC)+           ;INCREMENT ONE TIME GATE
1739  005154  177777                          TOFLAG: .WORD   -1              ;ONE TIME ENTANCE FLAG
1740  005156  001401                                  BEQ     10$             ;BRANCH IF FLAG IS NOW ZERO
```

```
1741  005160  000000                          HALT                    ;I HAVE ENTERED THIS ROUTINE BEFORE
1742                                                                  ;I FINISHED REPORTING THE FIRST ERROR
1743                                                                  ;THE SECOND ENTRY ADDRESS IS ON THE
1744                                                                  ;STACK AND THE FIRST ERROR CONDITION
1745                                                                  ;IS PROBABLY STILL LOCKED UP .
1746  005162  012637  001304          10$:   MOV     (KSP)+,OLDPC    ;SAVE RETURN ADDRESS
1747  005166  012637  001306                 MOV     (KSP)+,OLDPS    ;SAVE OLD PSW
1748  005172  013737  177766  001266         MOV     CPUERR,PCPUER   ;SAVE CPU ERROR REGISTER
1749  005200  013737  001266  177766         MOV     PCPUER,CPUERR   ;CLEAR CPU ERROR REGISTER
1750  005206  023737  001266  001264         CMP     PCPUER,CPUEXP   ;SEE IF EXPECTEED CONDITION CAME UP.
1751  005214  001402                         BEQ     1$              ;BRANCH IF IT WAS A TIMEOUT
1752  005216  104001                         ERROR   1               ;NOT RIGHT CONDITION
1753  005220  000414                         BR      3$              ;BRANCH TO EXIT
1754  005222  050037  001226          1$:    BIS     R0,ADDROR       ;PERFORM LOGIAL OR OF FAILING ADDRESS
1755  005226  005100                         COM     R0              ;GET R0 READY FOR AND
1756  005230  040037  001224                 BIC     R0,ADRAND               ;PERFORM LOGICAL AND
1757  005234  005100                         COM     R0              ;PUT R0 BACK AS IT WAS
1758  005236  005737  001254                 TST     ERRCNT          ;IS THIS THE FIRST ERROR
1759  005242  001002                         BNE     2$              ;BRANCH IF NOT FIRST ERROR
1760  005244  104201                         ERROR   201             ;NO REGISTER RESPONSE.
1761  005246  000401                         BR      3$              ;BRANCH TO EXIT
1762  005250  104301                  2$:    ERROR   301             ;CONTINUE NO RESPONSE TABLE
1763  005252  012737  177777  005154  3$:    MOV     #-1,TOFLAG      ;RESET ONE TIME GATE
1764  005260  013746  001306                 MOV     OLDPS,-(KSP)    ;RESTORE OLD PSW
1765  005264  013746  001304                 MOV     OLDPC,-(KSP)    ;PUSH RETURN ADDRESS BACK ON THE STACK
1766  005270  000006                         RTT                     ;RETURN TO THE TEST
1767
1768                                  .SBTTL  SUBROUTINE TO REPORT MAP REGISTERS THAT WILL NOT HOLD ZERO
1769                                  ;;**********************************************************************
1770                                  ;*
1771                                  ;*      THIS SUBROUTINE IS CALLED EVERY TIME A MAP REGISTER IS CLEARED
1772                                  ;*      AND FOUND TO NOT BE ZERO.  IT KEEPS A LOGICAL "AND" AND "OR" OF
1773                                  ;*      THE FAILING REGISTER'S ADDRESS AND DATA AND REPORTS ALL ERRORS.
1774                                  ;*      AT THE END OF THE TEST A SUMMARY ERROR MESSAGE IS GIVEN WHICH
1775                                  ;*      WILL REPORT THE LOGICAL "AND" AND "OR" OF THE ADDRESSES AND DATA.
1776                                  ;*
1777                                  ;;**********************************************************************
1778  005272                          WRITEERROR:
1779  005272  012637  001304                 MOV     (KSP)+,OLDPC    ;SAVE RETURN ADDRESS IN CASE OF ERROR LOOP
1780  005276  050037  001226                 BIS     R0,ADDROR       ;LOGICAL "OR" OF BAD ADDRESS
1781  005302  005100                         COM     R0              ;GET R0 READY FOR AND
1782  005304  040037  001224                 BIC     R0,ADRAND               ;PERFORM LOGICAL AND
1783  005310  005100                         COM     R0              ;PUT R0 BACK AS IT WAS
1784  005312  053737  001170  001232         BIS     $TMP0,DATAOR    ;LOGICAL "OR" OF BAD DATA
1785  005320  005137  001170                 COM     $TMP0           ;GET $TMP0 READY FOR AND
1786  005324  043737  001170  001230         BIC     $TMP0,DATAND            ;PERFORM LOGICAL AND
1787  005332  005137  001170                 COM     $TMP0           ;PUT $TMP0 BACK AS IT WAS
1788  005336  005737  001254                 TST     ERRCNT          ;SEE IF THIS IS FIRST ERROR
1789  005342  001002                         BNE     1$              ;BRANCH IF NOT FIRST ERROR
1790  005344  104202                         ERROR   202             ;ERROR TYPE OUT ITEM 4
1791  005346  000401                         BR      2$              ;SKIP NEXT STATEMENT
1792  005350  104302                  1$:    ERROR   302             ;ERROR TYPE OUT ITEM 5
1793  005352  000177  173726          2$:    JMP     @OLDPC          ;RETURN TO THE TEST
1794
1795                                  .SBTTL  SUBROUTINE TO REPORT DUAL ADDRESSING WHEN LOADING A MAP REGISTER
1796                                  ;;**********************************************************************
```

```
1797                            ;*
1798                            ;*      THIS SUBROUTINE WILL LOG AND REPORT ALL DUAL ADDRESSING ERRORS
1799                            ;*      FOUND IN THE MAPPING REGISTERS.  A 'LOGICAL OR'' AND A
1800                            ;*      'LOGICAL AND'' OF THE WRITTEN ADDRESSES WILL BE MAINTAINED IN
1801                            ;*      'ADDROR' AND 'ADRAND'.  THE LOG ON THE ADDITIONAL OR FAILING,
1802                            ;*      ADDRESSES WILL BE MAINTAINED IN 'DATAOR' AND 'DATAND'.
1803                            ;*
1804                            ;*********************************************************************
1805  005356                    DUALADR:                        ;STARTING LOCATION OF SUBROUTINE
1806  005356  012637  001304           MOV     (KSP)+,OLDPC    ;SAVE RETURN ADDRESS IN CASE OF LOOP
1807  005362  050037  001226           BIS     R0,ADDROR       ;LOGICAL OR OF WRITTEN ADDRESS
1808  005366  005100                   COM     R0              ;GET R0 READY FOR AND
1809  005370  040037  001224           BIC     R0,ADRAND               ;PERFORM LOGICAL AND
1810  005374  005100                   COM     R0              ;PUT R0 BACK AS IT WAS
1811  005376  050137  001232           BIS     R1,DATAOR       ;LOGICAL OR OF DUALED ADDRESS
1812  005402  005101                   COM     R1              ;GET R1 READY FOR AND
1813  005404  040137  001230           BIC     R1,DATAND               ;PERFORM LOGICAL AND
1814  005410  005101                   COM     R1              ;PUT R1 BACK AS IT WAS
1815  005412  005737  001254           TST     ERRCNT          ;SEE IF THIS IS FIRST ERROR
1816  005416  001002                   BNE     1$              ;BRANCH IF NOT FIRST ERROR
1817  005420  104203                   ERROR   203
1818  005422  000401                   BR      2$              ;BRANCH TO EXIT
1819  005424  104303           1$:     ERROR   303
1820  005426  000177  173652   2$:     JMP     @OLDPC          ;RETURN TO TEST
1821
1822                            .SBTTL  SUBROUTINE TO REPORT COUNT PATTERN ERRORS IN MAP REGISTERS
1823                            ;*********************************************************************
1824                            ;*
1825                            ;*      THIS SUBROUTINE IS CALLED WHENEVER AN ERROR IS DISCOVERED IN
1826                            ;*      THE COUNT PATTERN IN THE MAP REGISTERS.  IT KEEPS THE LOGICAL
1827                            ;*      ''AND'' AND 'OR'' OF THE FAILING REGISTER'S ADDRESS, DATA RECEIVED,
1828                            ;*      AND PATTERN LOADED.  EVERY ERROR IS REPORTED AND AT THE END OF
1829                            ;*      A TEST AN ERROR SUMMARY IS GIVEN.
1830                            ;*
1831                            ;*********************************************************************
1832  005432  012637  001304   COUNT.  MOV     (KSP)+,OLDPC    ;SAVE RETURN ADDRESS IN CASE OF ERROR LOOP
1833  005436  050037  001226           BIS     R0,ADDROR       ;LOGICAL OR OF BAD ADDRESSES
1834  005442  005100                   COM     R0              ;GET R0 READY FOR AND
1835  005444  040037  001224           BIC     R0,ADRAND               ;PERFORM LOGICAL AND
1836  005450  005100                   COM     R0              ;PUT R0 BACK AS IT WAS
1837  005452  050337  001232           BIS     R3,DATAOR       ;LOGICAL OR OF BAD DATA
1838  005456  005103                   COM     R3              ;GET R3 READY FOR AND
1839  005460  040337  001230           BIC     R3,DATAND               ;PERFORM LOGICAL AND
1840  005464  005103                   COM     R3              ;PUT R3 BACK AS IT WAS
1841  005466  050437  001236           BIS     R4,PATTOR       ;LOGICAL OR OF PATTERN LOADED
1842  005472  005104                   COM     R4              ;GET R4 READY FOR AND
1843  005474  040437  001234           BIC     R4,PATAND               ;PERFORM LOGICAL AND
1844  005500  005104                   COM     R4              ;PUT R4 BACK AS IT WAS
1845  005502  005737  001254           TST     ERRCNT          ;IS THIS FIRST ERROR
1846  005506  001002                   BNE     1$              ;BRANCH IF NOT FIRST ERROR
1847  005510  104204                   ERROR   204             ;REPORT FIRST COUNT ERROR
1848  005512  000401                   BR      2$              ;SKIP NEXT STATEMENT
1849  005514  104304           1$:     ERROR   304             ;REPORT MORE DATA
1850  005516  000177  173562   2$:     JMP     @OLDPC          ;RETURN TO THE TEST
1851
1852
```

```
                        .SBTTL  SUBROUTINE TO REPORT COUNT PATTERN ERRORS ON UNIBUS DATA PATH
                        ;;*****************************************************************************
                        ;*
                        ;*      THIS SUBROUTINE IS CALLED WHENEVER AN ERROR IS DISCOVERED IN THE
                        ;*      COUNT PATTERN THAT IS RUN OVER THE UNIBUS INTO MAIN MEMORY.  IT
                        ;*      KEEPS THE LOGICAL ''AND'' AND ''OR'' OF THE PATTERN LOADED AND THE
                        ;*      DATA RECEIVED, AND REPORTS ALL ERRORS.  AT THE END OF THE TEST
                        ;*      IT GIVES A SUMMARY MESSAGE OF ALL THE ERRORS.
                        ;*
                        ;;*****************************************************************************
005522  012637  001304  UBCOUNT:MOV     (KSP)+,OLDPC    ;SAVE RETURN ADDRESS IN CASE OF ERROR LOOP
005526  050137  001236          BIS     R1,PATTOR       ;LOGICAL OR OF COUNT LOADED
005532  005101                  COM     R1              ;GET R1 READY FOR AND
005534  040137  001234          BIC     R1,PATAND               ;PERFORM LOGICAL AND
005540  005101                  COM     R1              ;PUT R1 BACK AS IT WAS
005542  050037  001232          BIS     R0,DATAOR       ;LOGICAL OR OF DATA READ
005546  005100                  COM     R0              ;GET R0 READY FOR AND
005550  040037  001230          BIC     R0,DATAND               ;PERFORM LOGICAL AND
005554  005100                  COM     R0              ;PUT R0 BACK AS IT WAS
005556  005737  001254          TST     ERRCNT          ;IS THIS THE FIRST ERROR?
005562  001002                  BNE     1$              ;BRANCH IF NOT FIRST
005564  104205                  ERROR   205             ;REPORT FIRST ERROR ON UNIBUS DATA PATH
005566  000401                  BR      2$              ;SKIP NEXT INSTRUCTION
005570  104305          1$:     ERROR   305             ;REPORT MORE DATA FROM UNIBUS
005572  000177  173506  2$:     JMP     @OLDPC          ;RETURN TO THE TEST


                        .SBTTL  SUBROUTINE TO REPORT COUNT PATTERN ERRORS IN CACHE REGISTERS
                        ;;*****************************************************************************
                        ;*
                        ;*      THIS SUBROUTINE IS USED TO LOG AND REPORT THE COUNT PATTERN
                        ;*      ERRORS OCCURRING WHEN TESTING THE CACHE REGISTERS.
                        ;*      THE 'LOGICAL OR'' AND 'LOGICAL AND'' OF VARIOUS DATA WILL BE
                        ;*      MAINTAINED AS FOLLOWS:
                        ;*      DATA FETCHED FROM REGISTERS IN 'DATAOR' AND 'DATAND'
                        ;*      PATTERN LOADED INTO THE REGISTERS IN 'PATTOR' AND 'PATAND'.
                        ;*
                        ;;*****************************************************************************
005576                  CACOUNT:                        ;STARTING ADDRESS OF THIS SUBROUTINE
005576  012637  001304          MOV     (KSP)+,OLDPC    ;SAVE RETURN ADDRESS IN CASE OF LOOP
005602  050237  001236          BIS     R2,PATTOR       ;LOGICAL OR OF PATTERN LOADED
005606  005102                  COM     R2              ;GET R2 READY FOR AND
005610  040237  001234          BIC     R2,PATAND               ;PERFORM LOGICAL AND
005614  005102                  COM     R2              ;PUT R2 BACK AS IT WAS
005616  050337  001232          BIS     R3,DATAOR       ;LOGICAL OR OF DATA FETCHED
005622  005103                  COM     R3              ;GET R3 READY FOR AND
005624  040337  001230          BIC     R3,DATAND               ;PERFORM LOGICAL AND
005630  005103                  COM     R3              ;PUT R3 BACK AS IT WAS
005632  105737  001103          TSTB    $ERFLG          ;SEE IF THIS IS THE FIRST ERROR
005636  001002                  BNE     1$              ;BRANCH IF NOT FIRST ERROR
005640  104207                  ERROR   207
005642  000401                  BR      2$              ;BRANCH TO EXIT
005644  104307          1$:     ERROR   307
005646  000177  173432  2$:     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
                        ;*
                        ;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
005652  005227          CPUER:  INC     (PC)+           ;MAKE FLAG ZERO IF FIRST TIME
005654  177777          CPFLAG: .WORD   -1              ;NEGATIVE ONE FOR A FLAG
005656  001401                  BEQ     10$             ;BRANCH IF FIRST TIME IN
005660  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 .
005662  012637  001304  10$:    MOV     (KSP)+,OLDPC    ;SAVE RETURN ADDRESS IN CASE OF LOOP
005666  012637  001306          MOV     (KSP)+,OLDPS    ;SAVE OLD PSW IN CASE OF LOOP
005672  012706  001100          MOV     #KERSTK,KSP     ;MAKE SURE THAT THE KERNEL STACK POINTER
                                                        ;IS AT 1100.
005676  013737  177766  001266  MOV     CPUERR,PCPUER   ;SAVE CPU ERROR REGISTER
005704  013737  001304  001302  MOV     OLDPC,BADPC     ;SAVE PC+2 AT TIME OF ABORT
005712  005737  001264          TST     CPUFXP          ;SEE IF ANY CONDITION WAS EXPECTED
005716  001406                  BEQ     2$              ;BRANCH IF NO TRAP WAS EXPECTED
005720  023737  001266  001264  CMP     PCPUER,CPUEXP   ;SEE IF EXPECTED ERROR OCCURED
005726  001403                  BEQ     1$              ;BRANCH IF ERROR CODES MATCH
005730  104001                  ERROR   1               ;WRONG CPU TRAP CONDITION
005732  000401                  BR      1$              ;SKIP NEXT INSTRUCTION
005734  104002          2$:     ERROR   2               ;NO CPU TRAP EXPECTED
005736  013737  001266  177766  1$:     MOV     PCPUER,CPUERR   ;CLEAR CPU ERROR REGISTER
005744  012737  177777  005654          MOV     #-1,CPFLAG      ;MAKE FLAG NEGATIVE ONE FOR NEXT TIME
005752  013746  001306          MOV     OLDPS,-(KSP)    ;PUSH OLD PSW BACK ON STACK
005756  013746  001304          MOV     OLDPC,-(KSP)    ;PUSH RETURN ADDRESS BACK ON STACK
005762  000006                  RTT                     ;RETURN FROM INTERRUPT OR ABORT


                        .SBTTL  CACHE TRAPS AND ABORTS HANDLER ROUTINE
                        ;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        ;*
                        ;*      THIS SUBROUTINE WILL HANDLE ALL UNEXPECTED PARITY ERRORS.  IF
                        ;*      THE PARITY ERROR IS AN ABORT THE TEST THAT WAS RUNNING WILL
                        ;*      BE RESTARTED ONCE AFTER THE ABORT CONDITION IS REPORTED.  ON THE
                        ;*      SECOND ABORT IN A SINGLE TEST THE NEXT TEST IS ATTEMPTED
                        ;*      AFTER THE ABORT IS REPORTED.
                        ;*      IF THE PARITY ERROR IS A TRAP THE CACHE WILL BE CLEANED UP BY
                        ;*      REMOVING THE BAD WORD THAT MAY BE IN THE CACHE, AND THE TEST
                        ;*      WILL BE CONTINUED AFTER THE PARITY ERROR IS REPORTED.
```

```
                                        ;*
                                        ;**********************************************************************
                                MEMER:  INC     (PC)+           ;MAKE FLAG ZE O IF FIRST TIME
                                PAFLAG: .WORD   -1              ;NEGATIVE ONE FOR A FLAG
                                        BEQ     10$             ;BRANCH IF FIRST TIME IN
                                        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 .
005774  012737  006206  000114  10$:    MOV     #5$,CACHVEC     ;SET CACHE VECTOR TO RTI UNTIL THE
                                                                ;THE CACHE HAS BEEN FIXED.
006002  012637  001304          MOV     (KSP)+,OLDPC    ;SAVE RETURN ADDRESS IN CASE OF LOOP
006006  012637  001306          MOV     (KSP)+,OLDPS    ;SAVE OLD PSW IN CASE OF LOOP
006012  013737  177740  001270  MOV     @#LOADRS,PLOADR ;SAVE LOWER CACHE ADDR REG
006020  013737  177742  001272  MOV     @#HIADRS,PHIADR ;SAVE HIGH BITS OF FAILING ADDR
006026  013737  177744  001274  MOV     @#MEMERR,PPARER ;SAVE MEMORY ERROR REGISTER
006034  013737  177746  001276  MOV     @#CONTRL,PCONTR ;SAVE CONTROL REGISTER FOR TYPE OUT
006042  013737  177750  001300  MOV     @#MAINT,PMAINT  ;SAVE MAINTENENCE REGISTER
006050  013737  001304  001302  MOV     OLDPC,BADPC     ;SAVE PC+2 AT TIME OF ABORT
006056  032737  000014  001274  BIT     #14,PPARER      ;WAS THIS A MAIN MEMORY REFERENCE
006064  001005                  BNE     1$              ;BRANCH IF MAIN MEMORY PARITY ERROR
006066  012737  005764  000114  MOV     #MEMER,CACHVEC  ;LOAD ADDRESS OF THIS ROUTINE IN VECTOR
006074  104003                  ERROR   3               ;UNEXPECTED PARITY ABORT
006076  000415                  BR      2$              ;BRANCH TO SUBROUTINE EXIT
006100  010046          1$:     MOV     R0,-(KSP)       ;SAVE R0 ON STACK
006102  013700  001270          MOV     PLOADR,R0       ;PUT LOW 16 BITS OF BAD ADDR IN R0
006106  042700  176000          BIC     #176000,R0      ;CLEAR UPPER 6 BITS OF ADDRESS
006112  074027  000002          XOR     R0,#BIT1        ;REFERENCE OTHER WORD OF PAIR
006116  005710                  TST     (R0)            ;READ AT SAME INDEX AS BAD WORD
006120  012600                  MOV     (KSP)+,R0       ;RESTORE R0 FROM STACK
006122  012737  005764  000114  MOV     #MEMER,CACHVEC  ;LOAD ADDRESS OF THIS ROUTINE IN VECTOR
006130  104004                  ERROR   4               ;MAIN MEMORY PARITY ERROR
006132  005737  001322          TST     RETRY           ;ARE YOU RETRYING THIS TEST?
006136  001006                  BNE     3$              ;BRANCH IF THIS IS SECOND TRY
006140  005237  001322          INC     RETRY           ;SET RETRY FLAG
006144  013737  001106  001304  MOV     $LPADR,OLDPC    ;RETURN TO START OF THIS TEST
006152  000403                  BR      4$              ;BRANCH TO EXIT
006154  013737  001324  001304  3$:     MOV     NXTTST,OLDPC    ;RETURN TO START OF NEXT TEST
                                                                ;SINCE THIS IS THE SECOND ABORT
006162  013737  001274  177744  4$:     MOV     PPARER,MEMERR   ;CLEAR MEMORY ERROR REGISTER
006170  012737  177777  005766          MOV     #-1,PAFLAG      ;RESTORE A NEGATIVE ONE FOR NEXT TIME
006176  013746  001306          MOV     OLDPS,-(KSP)    ;PUSH OLD PSW BACK ON STACK
006202  013746  001304          MOV     OLDPC,-(KSP)    ;PUSH RETURN ADDRESS BACK ON STACK
006206  000006          5$:     RTT                     ;RETURN FROM INTERRUPT.


                        .SBTTL  MEMORY MANAGEMENT TRAPS AND ABORTS HANDLER ROUTINE
                        ;**********************************************************************
                        ;*
                        ;*      THIS  ROUTINE WILL HANDLE ALL SPURIOUS MEMORY MANAGEMENT TRAPS
                        ;*      AND ABORTS.  IT WILL REPORT THE CONDITION OF ALL THE MEMORY
                        ;*      MAMAGEMENT STATUS REGISTERS, AND THEN RETURN TO THE TEST AND
                        ;*      TRY TO CONTINUE RUNNING.
                        ;*
```

```
                                ;;*****************************************************************************
006210  005227                  MMTRAP: INC     (PC)+           ;MAKE FLAG ZERO IF FIRST TIME
006212  177777                  MMFLAG: .WORD   -1              ;FLAG SHOULD BE NEG ONE
006214  001401                          BEQ     10$             ;BRANCH IF FIRST TIME INTO ROUTINE
006216  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 .
006220  011637  001302          10$:    MOV     (KSP),BADPC     ;SAVE PC AT TIME OF ABORT OR TRAP
006224  012637  001304                  MOV     (KSP)+,OLDPC    ;SAVE RETURN ADDRESS IN CASE OF LOOP
006230  012637  001306                  MOV     (KSP)+,OLDPS    ;SAVE OLD PSW IN CASE OF LOOP
006234  013737  177572  001312          MOV     @#MMR0,PMMR0    ;SAVE STATUS REGISTER
006242  013737  177574  001314          MOV     @#MMR1,PMMR1    ;SAVE AUTO INC/DEC REGISTER
006250  013737  177576  001316          MOV     @#MMR2,PMMR2    ;SAVE VIRTUAL ADDRESS REGISTER
006256  104005                          ERROR   5               ;UNEXPECTED M.M. ABORT OR TRAP
006260  042737  177776  177572  1$:     BIC     #177776,@#MMR0  ;CLEAR ALL BITS EXCEPT 0
006266  012737  177777  006212          MOV     #-1,MMFLAG      ;RESTORE A NEGATIVE ONE TO FLAG
006274  013746  001306                  MOV     OLDPS,-(KSP)    ;PUSH OLD PSW ONTO STACK
006300  013746  001304                  MOV     OLDPC,-(KSP)    ;PUSH RETURN ADDRESS ON STACK
006304  000006                          RTT                     ;RETURN TO MAIN PROGRAM


                                .SBTTL  ***************************************************************
                                .SBTTL  ********************* START OF TEST CODE **********************
        010000                  .=10000         ;START TEST CODE AT ADDRESS 10000 (2K)

010000                          START:
010000  012737  000340  177776          MOV     #340,@#PS       ;;LOCK OUT ALL INTERRUPTS
010006  012706  001100                  MOV     #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
010012  005026                          CLR     (R6)+           ;;CLEAR MEMORY LOCATION
010014  022706  001136                  CMP     #$TKS,R6        ;;DONE?
010020  001374                          BNE     .-6             ;;LOOP BACK IF NO
010022  012706  001100                  MOV     #STACK,SP       ;;SETUP THE STACK POINTER
010026  012737  002214  000020          MOV     #$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
010034  012737  000340  000022          MOV     #340,@#IOTVEC+2 ;;LEVEL 7
010042  012737  002534  000030          MOV     #$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
010050  012737  000340  000032          MOV     #340,@#EMTVEC+2 ;;LEVEL 7
010056  012737  004506  000034          MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
010064  012737  000340  000036          MOV     #340,@#TRAPVEC+2;LEVEL 7
010072  012737  004550  000024          MOV     #$PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
010100  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;LEVEL 7
010106  013737  024426  024420          MOV     $ENDCT,$EOPCT   ;;SETUP END-OF-PROGRAM COUNTER
010114  005037  001204                  CLR     $TIMES          ;;INITIALIZE NUMBER OF ITERATIONS
010120  005037  001206                  CLR     $ESCAPE         ;;CLEAR THE ESCAPE ON ERROR ADDRESS
010124  112737  000001  001115          MOVB    #1,$ERMAX       ;;ALLOW ONE ERROR PER TEST
010132  012737  024646  000014          MOV     #$RTRN,@#TBITVEC ;;SET 'T' BIT VECTOR TO $RTRN
010140  012737  000340  000016          MOV     #340,@#TBITVEC+2 ;;LEVEL 7
010146  012737  000002  024646          MOV     #RTI,$RTRN      ;;SET $RTRN TO A RTI
010154  012737  010202  000010          MOV     #65$,@#RESVEC   ;;TRY TO DO A RTT
010162  005046                          CLR     -(SP)           ;;DUMMY PS
010164  012746  010172                  MOV     #64$,-(SP)      ;;AND PC
010170  000006                          RTT                     ;;TRY THE RTT
010172  012737  000006  024646  64$:     MOV     #RTT,$RTRN      ;;RTT IS LEGAL--SET $RTRN TO A RTT
```

```
2077  010200  000402                        BR      66$
2078  010202  062706  000010                65$:    ADD     #10,SP          ;;RTT ILLEGAL--CLEAN OFF THE STACK
2079  010206  012737  000012  000010        66$:    MOV     #RESVEC+2,@#RESVEC ;;RESTORE TRAP CATCHER
2080  010214  005037  024654                        CLR     $TBIT           ;;CLEAR 'T' BIT SWITCH
2081  010220  012737  010220  001106                MOV     #.,$LPADR       ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
2082  010226  012737  010226  001110                MOV     #.,$LPERR       ;;SETUP THE ERROR LOOP ADDRESS
2083  010234  005227  177777                       INC     #-1             ;;FIRST TIME?
2084  010240  001025                        BNE     67$             ;;BRANCH IF NO
2085  010242  022737  024600  000042                CMP     #$ENDAD,@#42    ;;ACT-11?
2086  010250  001421                        BEQ     67$             ;;BRANCH IF YES
2087  010252  104400  010260                TYPE    ,68$            ;;TYPE ASCIZ STRING
2088  010256  000416                        BR      67$             ;;GET OVER THE ASCIZ
2089                                  ;;68$:  .ASCIZ  <CRLF>?CEKBFD  11/70 UNIBUS MAP?<CRLF>
2090  010314                          67$:
2091                                  ;;
2092                                  ;;*** TEST FOR VARIOUS KB11 PROCESSORS ***
2093                                  ;;
2094                                  ;;*THIS ROUTINE POLES THE RESULTS OF ATTEMPTS TO SET TO ONE
2095                                  ;;*CERTAIN CRITICAL BITS THAT ARE KNOWN TO BE OPERATIVE ON A KB11CM,
2096                                  ;;*OR KB11EM PROCESSOR. IF TWO OUT OF FOUR OF THE TESTS ARE
2097                                  ;;*POSITIVE THEN THE KB11CM OR KB11EM FLAG IS SET,IF LESS THAN TWO OF THE
2098                                  ;;*TESTS ARE POSITIVE THEN THE KB11E FLAG OR NO FLAG IS SET. THE DETERMINATION
2099                                  ;;*OF WHICH PAIR IS VALID IS BASED ON THE RESULTS OF EXECUTING AN MFPT OPCODE
2100                                  ;;*(OPCODE 7).  IF THIS INSTRUCTION TRAPS THIS IS AN KB11CM OR
2101                                  ;;*A PLAIN 1170 (KB11-B OR KB11-C).  IF THE INSTRUCTION DOES NOT TRAP THEN
2102                                  ;;*THIS IS A KB11-E OR KB11-EM.
2103                                  ;;
2104  010314  105037  001332          KBTST:  CLRB    @#KB11CM        ;RESET THE MP FLAG
2105  010320  005037  001330                  CLR     @#KB11E         ;CLEAR KB11E AND KB11EM FLAGS
2106  010324  012737  010562  000010          MOV     #MFPTTR,@#RESVEC ;SET UP TRAP ADDRESS FOR MFPT AT RESERV VECTOR
2107  010332  000007                          MFPT                    ;EXECUTE MFPT. WILL TRAP ON 1170 (KB11B/C) OR
                                                                      ;KB11CM (11/74                          )
2108
2109  010334  012737  000001  001330          MOV     #1,@#KB11E      ;HERE IF KB11E OR KB11EM. SET FLAG
2110  010342  005037  177750          T1:     CLR     @#MAINT         ;CLEAR THE MAINTENANCE REGISTER
2111  010346  005005                          CLR     R5              ;RESET THE TEST COUNTER
2112  010350  012700  177746                  MOV     #CONTRL,R0      ;GET THE ADDRESS OF...
2113  010354  012701  177750                  MOV     #MAINT,R1       ;CCR,MAINT,AND MAPH00...
2114  010360  012702  170202                  MOV     #MAPH00,R2      ;AND PLACE IN R0-R2
2115  010364  052710  040000                  BIS     #BIT14,(R0)     ;TRY TO SET IVSS BIT
2116  010370  032710  040000                  BIT     #BIT14,(R0)     ;DID IT SET?
2117  010374  001403                          BEQ     T2              ;NO,GO TO NEXT TEST
2118  010376  042710  040000                  BIC     #BIT14,(R0)     ;CLEAR IT.
2119  010402  005205                          INC     R5              ;TEST IS POSITIVE
2120  010404  052711  000001          T2:     BIS     #BIT0,(R1)      ;SET EDMA IN MAINT REGISTER
2121  010410  032711  000001                  BIT     #BIT0,(R1)
2122  010414  001410                          BEQ     T3
2123  010416  052710  004000                  BIS     #BIT11,(R0)     ;TRY TO SET DMMA IN CCR
2124  010422  032710  004000                  BIT     #BIT11,(R0)
2125  010426  001403                          BEQ     T3
2126  010430  042710  004000                  BIC     #BIT11,(R0)
2127  010434  005205                          INC     R5
2128  010436  042711  000001          T3:     BIC     #BIT0,(R1)      ;MAKE SURE EDMA IS CLEAR
2129  010442  052737  100000  172300          BIS     #BIT15,KIPDR0   ;TRY TO SET BYP ON A PDR
2130  010450  032737  100000  172300          BIT     #BIT15,KIPDR0
2131  010456  001404                          BEQ     T4
2132  010460  042737  100000  172300          BIC     #BIT15,KIPDR0
```

```
2133  010466  005205                        INC     R5
2134  010470  052712  100000        T4:     BIS     #BIT15,(R2)     ;TRY TO SET BYP ON UNIBUS MAP
2135  010474  032712  100000                BIT     #BIT15,(R2)
2136  010500  001403                        BEQ     T.END
2137  010502  042712  100000                BIC     #BIT15,(R2)
2138  010506  005205                        INC     R5
2139  010510  022705  000002        T.END:  CMP     #2,R5           ;IS THE RESULT OF THE TEST >=2
2140  010514  101021                        BHI     2$              ;NO,THIS IT A KB11E OR KB11-B/C (11/70)
2141  010516  005000                        CLR     R0
2142  010520  005037  177746              CLR     @#CONTRL
2143  010524  013701  177746        3$:     MOV     @#CONTRL,R1
2144  010530  001402                        BEQ     4$
2145  010532  005200                        INC     R0
2146  010534  001373                        BNE     3$
2147  010536                        4$:
2148  010536  005737  001330              TST     @#KB11E         ;IS IS A KB11-E OR KB11-EM?
2149  010542  001404                        BEQ     1$              ;BR IF NEITHER. MUST BE KB11CM
2150  010544  012737  000400  001330      MOV     #BIT8,@#KB11E   ;SET UPPER BYTE (KB11-EM)
2151  010552  000402                        BR      2$              ;DONE
2152  010554  105237  001332        1$:     INCB    @#KB11CM        ;YES, FLAG THIS AS A MODIFIED PROCESSOR
2153  010560  000403                2$:     BR      ENDKB           ;DONE DETERMINING WHICH CPU
2154
2155  010562                        MFPTTR:                         ;HERE IF MFPT TRAPPED. SEE IF 1170 OR KB11CM
2156  010562  012716  010342              MOV     #T1,(SP)        ;SET UP RETURN ADDRESS FOR RTI
2157  010566  000002                        RTI                     ;RETURN
2158  010570                        ENDKB:
2159  010570  005227  177777              INC     #-1             ;FIRST TIME?
2160  010574  001026                        BNE     100$            ;BR IF NO
2161  010576  104400  024662              TYPE    ,MSG1           ;<15><12>CPU UNDER TEST FOUND TO BE A
2162  010602  005737  001330              TST     @#KB11E         ;IS THIS A KB11-E OR KB11-EM?
2163  010606  001011                        BNE     101$            ;BR IF EITHER ONE
2164  010610  105737  001332              TSTB    @#KB11CM        ;IS IT A 11/74          (KB11CM)
2165  010614  001003                        BNE     1$              ;BR IF IT IS
2166  010616  104400  024732              TYPE    ,MSG3           ;KB11-B/C<15><12>
2167  010622  000413                        BR      100$            ;SKIP OTHER MESSAGE
2168  010624  104400  024744        1$:     TYPE    ,MSG4           ;11/74          (KB11CM)<15><12>
2169  010630  000410                        BR      100$            ;SKIP CISP MESSAGE
2170  010632  105737  001330        101$:   TSTB    @#KB11E         ;IS IT A KB11-E?
2171  010636  001403                        BEQ     102$            ;BR IF NOT. MUST BE KB11-EM
2172  010640  104400  024775              TYPE    ,MSG5           ;KB11-E<15><12>
2173  010644  000402                        BR      100$            ;SKIP KB11-EM MESSAGE
2174  010646  104400  024721        102$:   TYPE    ,MSG2           ;KB11-EM<15><12>
2175  010652                        100$:
2176  010652  012737  005764  000114  LOOP:   MOV     #MEMER,CACHVEC  ;LOAD PARITY ERROR SERVICE
2177                                                                ;ROUTINE ADDRESS
2178  010660  012737  000340  000116        MOV     #340,CACHVEC+2  ;SET PRIORITY SEVEN
2179  010666  012737  006210  000250        MOV     #MMTRAP,MMVEC   ;LOAD MEMORY MANAGEMENT TRAP
2180                                                                ;SERVICE ROUTINE ADDRESS
2181  010674  012737  000340  000252        MOV     #340,MMVEC+2    ;SET PRIORITY SEVEN
2182  010702  012737  005652  000004        MOV     #CPUER,ERRVEC   ;LOAD CPU TRAP SERVICE ROUTINE ADDR
2183  010710  012737  000340  000006        MOV     #340,ERRVEC+2   ;SET PRIORITY SEVEN
2184  010716  012727  000044  177770        MOV     #044,#177770    ;LOAD ROM ADDRESS OF "NOP" INTO MICRO
2185                                                                ;BREAK REGISTER.  EVERY "NOP" WILL
2186                                                                ;GENERATE A "SYNC" PULSE ON PIN
2187                                                                ;#AE1 SLOT 10
2188  010724  005037  001264              CLR     CPUEXP          ;NOT EXPECTING ANY CPU ERRORS
```

```
                        ;********************* START OF TEST CODE ************************
2189  010730  005037  177572          CLR     @#MMR0          ;START IN 16 BIT MAPPING
2190  010734  005037  172516          CLR     @#MMR3          ;DISABLE MAP AND 22-BIT MAPPING
2191  010740  012700  177777          MOV     #-1,R0          ;NEGATIVE ONE USED TO INITIALIZE FLAGS
2192  010744  010037  005654          MOV     R0,CPFLAG       ;INITIALIZE FLAGS
2193  010750  010037  005154          MOV     R0,TOFLAG       ;INITIALIZE FLAGS
2194  010754  010037  005766          MOV     R0,PAFLAG       ;INITIALIZE FLAGS
2195  010760  010037  006212          MOV     R0,MMFLAG       ;INITIALIZE FLAGS
2196  010764  010037  177744          MOV     R0,@#MEMERR     ;CLEAR PARITY ERROR REGISTER
2197  010770  010037  177766          MOV     R0,@#CPUERR     ;CLEAR CPU ERROR REGISTER
2198
2199                                ;*THE FOLLOWING TWO TESTS ARE USED TO VERIFY THAT SOMETHING
2200                                ;*RESPONDS WHEN THE MAP REGISTERS AND CACHE REGISTERS ARE
2201                                ;*REFERENCED.  THE SIGNALS THAT SHOULD BE GENERATED ARE:
2202                                ;*'MAPB REG OP H' AND 'MAPB CACHE REG H'. UNDER PROPER
2203                                ;*OPERATION EITHER SIGNAL SHOULD CAUSE 'BUS SSYN L' TO BE
2204                                ;*ASSERTED AND NO TIMEOUT WILL OCCUR.
2205
2206                                ;;*****************************************************************
2207                                ;*TEST 1         MAP REGISTER RESPONSE TEST
2208                                ;*
2209                                ;*      THIS TEST IS USED TO ENSURE THAT ALL THE UNIBUS MAP REGISTERS
2210                                ;*      CAN BE REFERENCED UNDER PROGRAM CONTROL, WITHOUT TIMING OUT.
2211                                ;*      THE ADDRESSES OF ANY MAP REGISTERS THAT TIME OUT WILL BE REPORTED
2212                                ;*      AND, AT THE END OF THE TEST, A SUMMARY OF THOSE REGISTERS WILL
2213                                ;*      BE GIVEN.  THE ADDRESS DECODE CIRCUITRY IS ON 'MAPB' AND 'MAPC'.
2214                                ;*
2215                                ;;*****************************************************************
2216  010774                        TST1:
2217  010774  000004                       SCOPE
2218  010776  012737  011060  001324       MOV     #TST2,NXTTST    ;SAVE STARTING ADDRESS OF NEXT TEST
2219                                                                ;FOR ESCAPE ON PARITY ERRORS
2220  011004  012737  005152  000004 20$:   MOV     #TIMEOUT,ERRVEC ;LOAD ERRVEC WITH ROUTINE ADDR.
2221  011012  012700  170200               MOV     #MAPL0,R0       ;PUT FIRST MAP REG ADDR IN R0
2222  011016  012737  011024  001110       MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
2223  011024  000240                1$:    NOP                     ;THIS IS A SYNC POINT FOR SCOPING
2224  011026  011002                       MOV     (R0),R2         ;READ MAP REGISTERS TO R2
2225  011030  062700  000002               ADD     #2,R0           ;POINT TONEXT MAP REGISTER
2226  011034  022700  170376               CMP     #MAPH37,R0      ;COMPARE LAST MAP REG ADDR WITH R0
2227  011040  103371                       BHIS    1$              ;CONTINUE READING IF NOT AT LAST REG.
2228  011042  012737  011004  001110       MOV     #20$,$LPERR     ;INITIALIZE LOOPING POINTER IN CASE
2229                                                                ;OF INTERMITTENT FAULTS.
2230  011050  005737  001254               TST     ERRCNT          ;SEE IF THERE WERE ANY ERRORS
2231  011054  001401                       BEQ     TST2            ;;NO ERRORS GO TO NEXT TEST
2232  011056  104006                       ERROR   6               ;SUMMARY OF MAP REGISTERS THAT TIMED OUT
2233
2234                                ;;*****************************************************************
2235                                ;*TEST 2         CACHE REGISTER RESPONSE TEST
2236
2237                                ;*
2238                                ;*      THIS TEST IS USED TO ENSURE THAT ALL THE CACHE REGISTERS
2239                                ;*      CAN BE REFERENCED UNDER PROGRAM CONTROL, WITHOUT TIMING OUT.
2240                                ;*      THE ADDRESSES OF ANY CACHE REGISTERS THAT TIME OUT WILL BE
2241                                ;*      REPORTED AND, AT THE END OF THE TEST, A SUMMARY OF THOSE REGISTERS
2242                                ;*      WILL BE GIVEN.
2243                                ;*      THESE REGISTERS ARE TESTED HERE BECAUSE THE ADDRESS DECODE AND DATA
2244                                ;*      PATH IS ON THE UNIBUS MAP BOARD (8141).  THE ADDRESS DECODE CIRCUITRY
```

```
                                        T2      CACHE REGISTER RESPONSE TEST

                                        ;*      IS ON 'MAPB' AND GENERATES 'MAPB CACHE REG'.
                                        ;*
                                        ;*********************************************************************
                                        TST2:
                                                SCOPE
011060  000004                                 MOV     #TST3,NXTTST    ;SAVE STARTING ADDRESS OF NEXT TEST
011062  012737  011136  001324                                          ;FOR ESCAPE ON PARITY ERRORS
                                        20$:    MOV     #LOADRS,R0      ;PUT ADDR OF FIRST CACHE REG IN R0
011070  012700  177740                  MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER HERE
011074  012737  011102  001110          NOP                     ;THIS IS A SYNC POINT FOR SCOPING
011102  000240                  1$:     MOV     (R0),R2         ;TRY TO READ ALL CACHE REGISTERS
011104  011002                          ADD     #2,R0           ;POINT TO NEXT CACHE REGISTER
011106  062700  000002                  CMP     #HITMIS,R0      ;HAVE ALL REGISTERS BEEN REFERENCED?
011112  022700  177752                  BHIS    1$              ;BRANCH IF MORE TO TEST
011116  103371                          MOV     #20$,$LPERR     ;INITIALIZE LOOPING ERROR IN CASE
011120  012737  011070  001110                                          ;OF INTERMITTENT FAULTS
                                        TST     ERRCNT          ;WERE THERE ANY ERRORS.
011126  005737  001254                  BEQ     TST3            ;;BRANCH IF NO ERRORS
011132  001401                          ERROR   7               ;SUMMARY OF CACHE REGISTERS THAT TIMED OUT
011134  104007

                                        ;*THE NEXT FOUR (4) TESTS ARE USED TO CHECK THE GENERATION
                                        ;*OF: 'MAPB WRITE HI WORD L' AND 'MAPB WRITE LOWORD L'.
                                        ;*ALL REGISTERS ARE CLEARED AND THEN READ AND CHECKED FOR
                                        ;*ZEROES.  THESE TESTS ALSO DETECT BITS IN THE DATA PATH
                                        ;*TO AND FROM THE MAP REGISTERS OR IN THE MAP REGISTERS STUCK
                                        ;*AT '1'.  THE RECEIVERS ON PAGE MAPA AND THE B & C INPUTS
                                        ;*TO THE MULTIPLEXERS ON PAGE MAPJ AND THE DRIVERS ON PAGE
                                        ;*MAPJ MAKE UP THE DATA PATH UNDER TEST HERE.

                                        ;*********************************************************************
                                        ;*TEST 3        CLEAR AND READ LOW 20 REGISTERS LOWER 16 BITS
                                        ;*
                                        ;*      THIS TEST WILL ENSURE THAT MAPL00 - MAPL17 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 MAPL00 - MAPL17 WILL
                                        ;*      GIVE MORE HELP.
                                        ;*
                                        ;*********************************************************************
011174                                  TST3:
                                                SCOPE
011136  000004                                 MOV     #TST4,NXTTST    ;SAVE STARTING ADDRESS OF NEXT TEST
011140  012737  011234  001324                                          ;FOR ESCAPE ON PARITY ERRORS
                                                MOV     #CPUER,ERRVEC   ;LOAD CPU TRAP SERVICE ROUTINE ADDR
011146  012737  005652  000004          20$:    MOV     #MAPL0,R0       ;LOAD STARTING MAP REGISTER ADDR
011154  012700  170200                  MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER HERE
011160  012737  011166  001110          1$:     NOP                     ;THIS IS A SYNC POINT FOR SCOPING
011166  000240                          CLR     (R0)            ;CLEAR MAP REGISTER
011170  005010                          MOV     (R0),$TMP0      ;SEE IF MAP REGISTER IS ZERO
011172  011037  001170                  BEQ     2$              ;BRANCH IF REGISTER IS ZERO
011176  001402                          JSR     PC,WRITEERROR   ;REPORT AND LOG ERROR
011200  004737  005272          2$:     ADD     #4,R0           ;POINT TO NEXT MAP REG LOWER 16 BITS
011204  062700  000004                  CMP     #MAPL17,R0      ;SEE IF THIS SECTION IS TESTED
011210  022700  170274                  BHIS    1$              ;BRANCH IF NOT
011214  103364
```

```
2301  011216  012737  011154  001110          MOV     #20$,$LPERR     ;INITIALIZE LOOPING POINTER IN CASE
2302                                                                    ;OF INTERMITTENT FAULTS.
2303  011224  005737  001254                  TST     ERRCNT          ;WERE THERE ANY ERRORS ON THIS TEST
2304  011230  001401                          BEQ     TST4            ;;BRANCH IF NO ERRORS
2305  011232  104010                          ERROR   10              ;SUMMARY OF MAP REGS THAT COULD NOT
2306                                                                    ;HOLD ZERO
2307
2308
2309                                  ;;*****************************************************************
2310                                  ;*TEST 4        CLEAR AND READ LOW 20 REGISTERS UPPER 6 BITS
2311                                  ;*
2312                                  ;*      THIS TEST WILL ENSURE THAT MAPH00 - MAPH17 WILL ALL HOLD ZERO.
2313                                  ;*      IF ANY REGISTERS ARE NOT ZERO AFTER BEING CLEARED, THEIR ADDRESS AND
2314                                  ;*      CONTENTS ARE REPORTED.  IF ALL THE REGISTERS HAVE RANDOM DATA THEN
2315                                  ;*      PROBABLY NO WRITE PULSE IS BEING GENERATED, BUT IF THE ERROR IS IN
2316                                  ;*      ONLY ONE REGISTER OR IN ONE BIT IT MIGHT BE MORE DIFFICULT TO FIND.
2317                                  ;*      IT IS POSSIBLE THAT THE COUNT PATTERN TEST FOR MAPH00 - MAPH17 WILL
2318                                  ;*      GIVE MORE HELP.
2319                                  ;*
2320                                  ;*****************************************************************
2321  011234                          TST4:
2322  011234  000004                          SCOPE
2323  011236  012737  011324  001324          MOV     #TST5,NXTTST    ;SAVE STARTING ADDRESS OF NEXT TEST
2324                                                                    ;FOR ESCAPE ON PARITY ERRORS
2325  011244  012700  170202          20$:    MOV     #MAPH0,R0       ;LOAD STARTING MAP REGISTER ADDR
2326  011250  012737  011256  001110          MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER HERE
2327  011256  000240                  1$:     NOP                     ;THIS IS A SYNC POINT FOR SCOPING
2328  011260  005010                          CLR     (R0)            ;CLEAR MAP REGISTER
2329  011262  011037  001170                  MOV     (R0),$TMP0      ;SEE IF MAP REGISTER IS ZERO
2330  011266  001402                          BEQ     2$              ;BRANCH IF REGISTER IS ZERO
2331  011270  004737  005272                  JSR     PC,WRITEERROR   ;REPORT AND LOG ERROR
2332  011274  062700  000004          2$:     ADD     #4,R0           ;POINT TO NEXT MAP REG UPPER 6 BITS
2333  011300  022700  170276                  CMP     #MAPH17,R0      ;SEE IF TESTING IS OVER
2334  011304  103364                          BHIS    1$              ;BRANCH IF MORE REGS TO TEST
2335  011306  012737  011244  001110          MOV     #20$,$LPERR     ;INITIALIZE LOOPING POINTER IN CASE
2336                                                                    ;OF INTERMITTENT FAULTS.
2337  011314  005737  001254                  TST     ERRCNT          ;WHERE THERE ANY ERRORS ON THIS
2338  011320  001401                          BEQ     TST5            ;;BRANCH IF NO ERRORS
2339  011322  104011                          ERROR   11              ;SUMMARY OF MAP REGS THAT COULD NOT
2340                                                                    ;HOLD ZERO
2341
2342
2343                                  ;;*****************************************************************
2344                                  ;*TEST 5        CLEAR AND READ HIGH 20 REGISTERS LOWER 16 BITS
2345                                  ;*
2346                                  ;*      THIS TEST WILL ENSURE THAT MAPL20 - MAPL37 WILL ALL HOLD ZERO.
2347                                  ;*      IF ANY REGISTERS ARE NOT ZERO AFTER BEING CLEARED, THEIR ADDRESS AND
2348                                  ;*      CONTENTS ARE REPORTED.  IF ALL THE REGISTERS HAVE RANDOM DATA THEN
2349                                  ;*      PROBABLY NO WRITE PULSE IS BEING GENERATED, BUT IF THE ERROR IS IN
2350                                  ;*      ONLY ONE REGISTER OR IN ONE BIT IT MIGHT BE MORE DIFFICULT TO FIND.
2351                                  ;*      IT IS POSSIBLE THAT THE COUNT PATTERN TEST FOR MAPL20 - MAPL37 WILL
2352                                  ;*      GIVE MORE HELP.
2353                                  ;*
2354                                  ;*****************************************************************
2355  011324                          TST5:
2356  011324  000004                          SCOPE
```

```
2357  011326  012737  011414  001324          MOV     #TST6,NXTTST    ;SAVE STARTING ADDRESS OF NEXT TEST
2358                                                                  ;FOR ESCAPE ON PARITY ERRORS
2359  011334  012700  170300              20$:  MOV     #MAPL20,R0      ;LOAD STARTING MAP REGISTER ADDR
2360  011340  012737  011346  001110            MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER HERE
2361  011346  000240                      1$:   NOP                     ;THIS IS A SYNC POINT FOR SCOPING
2362  011350  005010                            CLR     (R0)            ;CLEAR MAP REGISTER
2363  011352  011037  001170                    MOV     (R0),$TMP0      ;SEE IF MAP REGISTER IS ZERO
2364  011356  001402                            BEQ     2$              ;BRANCH IF REGISTER IS ZERO
2365  011360  004737  005272                    JSR     PC,WRITEERROR   ;REPORT AND LOG ERROR
2366  011364  062700  000004              2$:   ADD     #4,R0           ;POINT TO NEXT REGISTER
2367  011370  022700  170374                    CMP     #MAPL37,R0      ;SEE IF THIS SECTION IS TESTED
2368  011374  103364                            BHIS    1$              ;BRANCH IF NOT
2369  011376  012737  011334  001110            MOV     #20$,$LPERR     ;INITIALIZE LOOPING POINTER IN CASE
2370                                                                    ;OF INTERMITTENT FAULTS.
2371  011404  005737  001254                    TST     ERRCNT          ;WERE THERE ANY ERRORS ON THIS
2372  011410  001401                            BEQ     TST6            ;;BRANCH IF NO ERRORS
2373  011412  104010                            ERROR   10              ;SUMMARY OF MAP REGS THAT COULD NOT
2374                                                                    ;HOLD ZERO
2375
2376
2377                          ;;*****************************************************************
2378                          ;*TEST 6        CLEAR AND READ HIGH 20 REGISTERS UPPER 6 BITS
2379                          ;*
2380                          ;*      THIS TEST WILL ENSURE THAT MAPH20 - MAPH37 WILL ALL HOLD ZERO.
2381                          ;*      IF ANY REGISTERS ARE NOT ZERO AFTER BEING CLEARED, THEIR ADDRESS AND
2382                          ;*      CONTENTS ARE REPORTED.  IF ALL THE REGISTERS HAVE RANDOM DATA THEN
2383                          ;*      PROBABLY NO WRITE PULSE IS BEING GENERATED, BUT IF THE ERROR IS IN
2384                          ;*      ONLY ONE REGISTER OR IN ONE BIT IT MIGHT BE MORE DIFFICULT TO FIND.
2385                          ;*      IT IS POSSIBLE THAT THE COUNT PATTERN TEST FOR MAPH20 - MAPH37 WILL
2386                          ;*      GIVE MORE HELP.
2387                          ;*
2388                          ;;*****************************************************************
2389  011414                  TST6:
2390  011414  000004                 SCOPE
2391  011416  012737  011504  001324          MOV     #TST7,NXTTST    ;SAVE STARTING ADDRESS OF NEXT TEST
2392                                                                  ;FOR ESCAPE ON PARITY ERRORS
2393  011424  012700  170302              20$:  MOV     #MAPH20,R0      ;LOAD STARTING MAP REGISTER ADDR
2394  011430  012737  011436  001110            MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER HERE
2395  011436  000240                      1$:   NOP                     ;THIS IS A SYNC POINT FOR SCOPING
2396  011440  005010                            CLR     (R0)            ;CLEAR MAP REGISTER
2397  011442  011037  001170                    MOV     (R0),$TMP0      ;SEE IF MAP REGISTER IS ZERO
2398  011446  001402                            BEQ     2$              ;BRANCH IF REGISTER IS ZERO
2399  011450  004737  005272                    JSR     PC,WRITEERROR   ;REPORT AND LOG ERROR
2400  011454  062700  000004              2$:   ADD     #4,R0           ;POINT TO NEXT HIGH REGISTER
2401  011460  022700  170376                    CMP     #MAPH37,R0      ;SEE IF THERE ARE MORE REGS LEFT
2402  011464  103364                            BHIS    1$              ;BRANCH IF MORE REGS TO TEST
2403  011466  012737  011424  001110            MOV     #20$,$LPERR     ;INITIALIZE LOOPING POINTER IN CASE
2404                                                                    ;OF INTERMITTENT FAULTS.
2405  011474  005737  001254                    TST     ERRCNT          ;WERE THERE ANY ERRORS ON THIS TEST
2406  011500  001401                            BEQ     TST7            ;;BRANCH IF NO ERRORS
2407  011502  104011                            ERROR   11              ;SUMMARY OF MAP REGS THAT COULD NOT
2408                                                                    ;HOLD ZERO
2409
2410                          ;*IN THE NEXT TWO TESTS A COUNT PATTERN IS RUN THROUGH
2411                          ;*MAP REGISTER 00 TO TEST THE DATA PATH TO AND FROM THE
2412                          ;*MAP REGISTERS.  THE DATA RECEIVERS ARE ON PAGE MAPA, AND
```

```
                                        ;*THE DATA DRIVERS ARE ON PAGE MAPJ.  THE BINPUTS TO THE
                                        ;*MULTIPLEXERS ON PAGE MAPJ ARE USED WHEN READING THE LOWER 16 BITS (TEST 7) AND
                                        ;*THE C INPUTS ARE USED WHEN READING THE UPPER 6 BITS OF THE
                                        ;*MAP REGISTER (TEST 10).  IF REGISTER 0 FAILS THEN REGISTER 20
                                        ;*IS TRIED AND THE ERROR IS REPORTED EVEN IF REGISTER 20 SUCCEEDS.

                                        ;*****************************************************************
                                        ;*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 MAPL00, 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'.
                                        ;*
                                        ;*****************************************************************
011504                                  TST7:
                                                SCOPE
011504  000004
011506  012737  011630  001324                    MOV     #TST10,NXTTST   ;SAVE STARTING ADDRESS OF NEXT TEST
                                                                        ;FOR ESCAPE ON PARITY ERRORS
011514  012737  000012  001204                    MOV     #12.,$TIMES     ;;DO 12 ITERATIONS
011522  005003                          20$:      CLR     R3              ;START COUNT PATTERN AT 0
011524  005002                                    CLR     R2              ;START COUNT PATTERN AT 0
011526  012737  011534  001110                    MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
011534  000240                          1$:       NOP                     ;THIS IS A SYNC POINT FOR SCOPING
011536  010237  170200                          MOV     R2,MAPL00       ;LOAD MAP REG 0 LOW 16 BITS
011542  013700  170200                          MOV     MAPL00,R0       ;READ CONTENTS OF MAP REGISTER 0
011546  020200                                  CMP     R2,R0           ;COMPARE DATA & PATTERN
011550  001402                                  BEQ     2$              ;BRANCH IF DATA DOES NOT MATCH
011552  104012                                  ERROR   12              ;POSSIBLE DATA PATH FAILURE
011554  000404                                  BR      3$              ;BRANCH TO TEST DATA PATH WITH
                                                                        ;A REGISTER OF THE NEXT GROUP
011556  062702  000002                  2$:     ADD     #2,R2           ;ADD 2 TO COUNT PATTERN
011560  001364                                  BNE     1$              ;BRANCH IF COUNT NOT COMPLETE.
011562  000416                                  BR      10$             ;DATA PATH OKAY FOR LOW 16 BITS
011564  012737  011574  001110          3$:     MOV     #12$,$LPERR     ;SET LOOP ON ERROR POINTER TO 12$
011574  000240                          12$:    NOP                     ;THIS IS A SYNC POINT FOR SCOPING
011576  010337  170300                          MOV     R3,MAPL20       ;LOAD MAP REG 20 LOW 16 BITS
011602  013701  170300                          MOV     MAPL20,R1       ;READ DATA STORED IN MAP REGISTER 20
011606  020301                                  CMP     R3,R1           ;COMPARE DATA & PATTERN
011610  001003                                  BNE     4$              ;CAUGHT ERROR,NOW SEE IF SAME.
011612  062703  000002                          ADD     #2,R3           ;ADD 2 TO COUNT PATTERN
011616  001357                                  BNE     2$              ;BRANCH IF COUNT NOT 0
011620  104013                          4$:     ERROR   13              ;PROBABLY IS A REAL DATA PATH ERROR
011622  012737  011522  001110          10$:    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 MAPH00, 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'.
                        ;*
                        ;**************************************************************
011630                  TST10:
                                SCOPE
011630  000004
011632  012737  012030  001324        MOV     #TST11,NXTTST   ;SAVE STARTING ADDRESS OF NEXT TEST
                                                                ;FOR ESCAPE ON PARITY ERRORS
011640  012737  000002  001204        MOV     #2,$TIMES       ;;DO 2 ITERATIONS
011646  005002                20$:    CLR     R2              ;START COUNT PATTERN AT 0
011650  005003                        CLR     R3              ;START COUNT PATTERN AT 0
011652  005004                        CLR     R4              ;START COUNT AT ZERO
011654  105737  001331                TSTB    KB11EM          ;IS THIS A KB11-EM?
011660  001003                        BNE     55$             ;BR IF IT IS
011662  105737  001332                TSTB    KB11CM          ;IS THIS A MODIFIED PROCESSOR?
011666  001403                        BEQ     56$             ;NO,CARRY ON
011670  012737  012026  012022  55$:  MOV     #72$+4,72$ ;USE THE MODIFIED MASK
011676  012737  011704  001110  56$:  MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
011704  000240                1$:     NOP                     ;THIS IS A SYNC POINT FOR SCOPING
011706  010402                        MOV     R4,R2           ;MOVE COUNT TO R2
011710  010237  170202                MOV     R2,@#MAPH0      ;LOAD MAP REG0 HIGH 6 BITS
011714  013700  170202                MOV     MAPH00,R0       ;READ BACK MAP REG 00 UPPER SIX BITS
011720  047702  000076                BIC     @72$,R2 ;       ;MASK OUT UNUSED BITS
011724  020200                        CMP     R2,R0           ;COMPARE DATA AND PATTERN LOADED
011726  001402                        BEQ     2$              ;BRANCH IF DATA IS CORRECT
011730  104012                        ERROR   12              ;POSSIBLE DATA PATH ERROR
011732  000405                        BR      3$              ;BRANCH TO TRY SECOND BANK OF MAP REGS
011734  005204                2$:     INC     R4              ;CHANGE DATA PATTERN
011736  022704  177777                CMP     #177777,R4      ;HAS COUNT REACH MAXIMUM?
011742  001360                        BNE     1$              ;BRANCH IF TEST NOT OVER
011744  000422                        BR      10$             ;UPPER SIX BIT DATA PATH IS OKAY
011746  012737  011754  001110  3$:   MOV     #12$,$LPERR     ;SET LOOP ON ERROR POINTER TO 12$
011754  000240                12$:    NOP                     ;THIS IS A SYNC POINT FOR SCOPING
011756  010403                        MOV     R4,R3
011760  010337  170302                MOV     R3,@#MAPH20     ;LOAD MAP REG 20 HIGH 6 BITS
011764  013701  170302                MOV     MAPH20,R1       ;READ BACK MAP REG 20 UPPER SIX BITS
011770  047703  000026                BIC     @72$,R3
011774  020301                        CMP     R3,R1           ;COMPARE DATA AND PATTERN
011776  001004                        BNE     4$              ;BRANCH IF DATA DOES NOT MATCH
012000  005204                        INC     R4              ;CHANGE DATA PATTERN
012002  022704  177777                CMP     #177777,R4      ;HAS COUNT REACHED MAXIMUM?
012006  001352                        BNE     2$              ;BRANCH IF TEST NOT OVER
012010  104013                4$:     ERROR   13              ;PROBABLY IS A REAL DATA PATH ERROR
012012  012737  011646  001110  10$.  MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
012020  000403                        BR      72$+6
012022  012024                72$:    .WORD   72$+2           ;POINTER TO PROPER MASK
012024  177700                        .WORD   177700          ;NONMODIFIED PROCESSORS USE THIS
012026  077700                        .WORD   077700          ;MODIFIED PROCESSORS USE THIS

                        ;;**************************************************************
                        ;*TEST 11        DUAL ADDRESSING ON LOADING AND READING MAP REGISTERS
                        ;*
                        ;*      THIS TEST ENSURES THAT ONLY ONE UNIBUS MAP REGISTER IS LOADED
```

```
2525                                    ;*      DURING A 'MOV   #DATA,@#MAPREG'' INSTRUCTION.  ALL MAP REGISTERS
2526                                    ;*      ARE CLEARED AND ONE REGISTER AT A TIME, STARTING WITH MAPL00,
2527                                    ;*      IS LOADED A NEGATIVE ONE.  THEN, ALL MAP REGISTERS ARE READ,
2528                                    ;*      STARTING WITH MAPH37, AND VERIFIED TO BE ZERO.  ANY REGISTER
2529                                    ;*      THAT IS NOT ZERO AND WHOSE UNIBUS ADDRESS DOES NOT MATCH THAT
2530                                    ;*      OF THE REGISTER UNDER TEST IS REPORTED TO BE IN ERROR.  AT THE
2531                                    ;*      END OF THE TEST A SUMMARY OF ALL DUALED REGISTERS IS GIVEN.
2532                                    ;*      THE SIGNALS THAT ARE TESTED HERE ARE ON 'MAPC' AND ARE:
2533                                    ;*      'EN LOREG'' & 'EN HIREG''.  THE ''A'' INPUTS TO THE ADDRESS
2534                                    ;*      MULTIPLEXER SHOULD BE THE ONES IN USE DURING A READ OR WRITE
2535                                    ;*      TO THE UNIBUS MAP REGISTERS.
2536                                    ;*
2537                                    ;**************************************************************************
2538  012030                            TST11:
2539  012030  000004                           SCOPE
2540  012032  012737  012154  001324           MOV     #TST12,NXTTST   ;SAVE STARTING ADDRESS OF NEXT TEST
2541                                                                    ;FOR ESCAPE ON PARITY ERRORS
2542  012040  012737  000144  001204           MOV     #144,$TIMES     ;;DO 144 ITERATIONS
2543  012046  012700  170200           20$:    MOV     #MAPL0,R0       ;LOAD FIRST MAP REG ADDR INTO R0
2544  012052  012737  012060  001110           MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER HERE
2545  012060  012701  170376           1$:     MOV     #MAPH37,R1      ;PUT ADDRESS OF HIGHEST MAP REG IN R1
2546  012064  004737  005134                   JSR     PC,CLRMAP       ;CLEAR ALL MAP REGISTERS IN CASE OF
2547                                                                    ;A 'FLAKEY' PROBLEM
2548  012070  000240                           NOP                     ;THIS IS A SYNC POINT FOR SCOPING
2549  012072  012710  177777                   MOV     #177777,(R0)    ;LOAD MAP REGISTER UNDER TEST
2550  012076  011102                   2$:     MOV     (R1),R2         ;READ MAP REGS STARTING FROM TOP
2551  012100  001404                           BEQ     3$              ;GO TO 3$ IF ZERO
2552  012102  020100                           CMP     R1,R0           ;SEE IF NON-ZERO ADDRS MATCH
2553  012104  001402                           BEQ     3$              ;GO TO 3$ IF MATCH
2554  012106  004737  005356                   JSR     PC,DUALADR      ;JUMP TO SUBROUTINE TO LOG ALL ERRORS
2555  012112  162701  000002           3$:     SUB     #2,R1           ;POINT TO NEXT MAP REGISTER
2556  012116  022701  170200                   CMP     #MAPL0,R1       ;HAVE ALL REGISTERS BEEN READ
2557  012122  101765                           BLOS    2$              ;IF NOT CONTINUE TEST
2558  012124  062700  000002                   ADD     #2,R0           ;POINT TO NEXT MAP REGISTER
2559  012130  022700  170376                   CMP     #MAPH37,R0      ;SEE IF TEST IS OVER
2560  012134  103351                           BHIS    1$              ;CONTINUE TESTING
2561  012136  012737  012046  001110           MOV     #20$,$LPERR     ;INITIALIZE LOOPING POINTER IN CASE
2562                                                                    ;OF INTERMITTENT FAULTS.
2563  012144  005737  001254                   TST     ERRCNT          ;SEE IF THERE WERE ANY ERRORS
2564  012150  001401                           BEQ     TST12           ;;GOTO NEXT TEST IF NO ERRORS
2565  012152  104014                           ERROR   14              ;SUMMARY OF DUAL ADDRESSING ERRORS
2566                                                                    ;ON LOADING MAP REGISTERS
2567
2568
2569                                    ;*THE NEXT FOUR(4) TESTS RUN A COUNT PATTERN THROUGH EACH
2570                                    ;*MAP REGISTER, CHECKING FOR STUCK BITS, AND SHORTED PINS.
2571
2572                                    ;**************************************************************************
2573                                    ;*TEST 12        COUNT PATTERN LOW 20 MAP REGISTERS LOWER 16 BITS
2574                                    ;*
2575                                    ;*      THIS TEST WILL RUN A COUNT PATTERN THROUGH UNIBUS MAP REGISTERS
2576                                    ;*      MAPL00 - MAPL17.  IF THE COUNT RECIEVED DOES NOT MATCH THE
2577                                    ;*      EXPECTED PATTERN THEN THE MAP REGISTER ADDRESS, DATA RECEIVED,
2578                                    ;*      PATTERN LOADED, AND PATTERN EXPECTED ARE REPORTED.  AT THE END
2579                                    ;*      OF THE TEST A SUMMARY OF ALL ERRORS IS GIVEN SO THAT YOU CAN
2580                                    ;*      DETERMINE IF THE ERROR IS BIT SENSITIVE OR REGISTER SENSITIVE.
```

```
                                        ;*      THE REGISTERS ARE ON 'MAPC' AND THE OUTPUT MULTIPLEXER IS ON
                                        ;*      'MAPJ'.
                                        ;*****************************************************************
                                        TST12:
012154                                          SCOPE
012154  000004
012156  012737  012300  001324                  MOV     #TST13,NXTTST   ;SAVE STARTING ADDRESS OF NEXT TEST
                                                                        ;FOR ESCAPE ON PARITY ERRORS
012164  012737  000012  001204                  MOV     #12.,$TIMES     ;;DO 12 ITERATIONS
012172  005002                          20$:    CLR     R2              ;START WITH AN ALL ZERO PATTERN
012174  012737  012214  001110                  MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
012202  010204                          1$:     MOV     R2,R4           ;PUT COUNT IN R4 FOR MASK
012204  042704  000001                          BIC     #000001,R4      ;CLEAR BITS IN MASK FOR PROPER COMPARE
012210  012700  170200                          MOV     #MAPL00,R0      ;LOAD STARTING MAP REGISTER ADDRESS
012214  000240                          2$:     NOP                     ;THIS IS A SYNC POINT FOR SCOPING
012216  010210                                  MOV     R2,(R0)         ;LOAD MAP REGISTER WITH COUNT PATTERN
012220  011003                                  MOV     (R0),R3         ;READ MAP REGISTER INTO R3
012222  020403                                  CMP     R4,R3           ;COMPARE MASK WITH DATA
012224  001402                                  BEQ     3$              ;BRANCH IF DATA MATCHES
012226  004737  005432                          JSR     PC,COUNT        ;JUMP TO COUNT TO LOG ALL ERRORS
012232  005003                          3$:     CLR     R3              ;CLEAR DATA HOLDER
012234  062700  000004                          ADD     #4,R0           ;POINT TO NEXT MAP REGISTER UNDER TEST
012240  022700  170274                          CMP     #MAPL17,R0      ;SEE IF ALL REGISTERS HAVE BEEN LOADED
012244  103363                                  BHIS    2$              ;BRANCH IF STILL MORE TO GO
012246  022702  177777                          CMP     #177777,R2      ;SEE IF COUNT HAS CYCLED
012252  001403                                  BEQ     4$              ;BRANCH IF TEST IS DONE
012254  062702  000401                          ADD     #401,R2         ;CHANGE COUNT PATTERN
012260  000750                                  BR      1$              ;CONTINUE TESTING
012262  012737  012172  001110          4$:     MOV     #20$,$LPERR     ;INITIALIZE LOOPING POINTER IN CASE
                                                                        ;OF INTERMITTENT FAULTS.
012270  005737  001254                          TST     ERRCNT          ;SEE IF THERE WERE ANY ERRORS
012274  001401                                  BEQ     TST13           ;;BRANCH IF NO ERRORS
012276  104000                                  ERROR                   ;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:
012300                                          SCOPE
012300  000004
012302  012737  012424  001324                  MOV     #TST14,NXTTST   ;SAVE STARTING ADDRESS OF NEXT TEST
                                                                        ;FOR ESCAPE ON PARITY ERRORS
012310  012737  000144  001204                  MOV     #144,$TIMES     ;;DO 144 ITERATIONS
012316  005002                          20$:    CLR     R2              ;START WITH AN ALL ZERO PATTERN
012320  012737  012340  001110                  MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
012326  010204                          1$:     MOV     R2,R4           ;PUT COUNT IN R4 FOR MASK
012330  042704  177700                          BIC     #177700,R4      ;CLEAR BITS IN MASK FOR PROPER COMPARE
```

```
2637  012334  012700  170202          MOV     #MAPH00,R0      ;LOAD STARTING MAP REGISTER ADDRESS
2638  012340  000240           2$:    NOP                     ;THIS IS A SYNC POINT FOR SCOPING
2639  012342  010210                  MOV     R2,(R0)         ;LOAD MAP REGISTER WITH COUNT PATTERN
2640  012344  011003                  MOV     (R0),R3         ;READ MAP REGISTER INTO R3
2641  012346  020403                  CMP     R2,R3           ;COMPARE MASK WITH DATA
2642  012350  001402                  BEQ     3$              ;BRANCH IF DATA MATCHES
2643  012352  004737  005432          JSR     PC,COUNT        ;JUMP TO COUNT TO LOG ALL ERRORS
2644  012356  005003           3$:    CLR     R3              ;CLEAR DATA HOLDER
2645  012360  062700  000004          ADD     #4,R0           ;POINT TO NEXT MAP REGISTER UNDER TEST
2646  012364  022700  170276          CMP     #MAPH17,R0      ;SEE IF ALL REGISTERS HAVE BEEN LOADED
2647  012370  103363                  BHIS    2$              ;BRANCH IF STILL MORE TO GO
2648  012372  022702  000177          CMP     #177,R2 ;SEE IF COUNT HAS CYCLED
2649  012376  001403                  BEQ     4$              ;BRANCH IF TEST IS DONE
2650  012400  062702  000001          ADD     #1,R2           ;CHANGE COUNT PATTERN
2651  012404  000750                  BR      1$              ;CONTINUE TESTING
2652  012406  012737  012316  001110 4$:    MOV     #20$,$LPERR     ;INITIALIZE LOOPING POINTER IN CASE
2653                                                          ;OF INTERMITTENT FAULTS.
2654  012414  005737  001254          TST     ERRCNT          ;SEE IF THERE WERE ANY ERRORS
2655  012420  001401                  BEQ     TST14           ;;BRANCH IF NO ERRORS
2656  012422  104016                  ERROR   16              ;SUMMARY OF COUNT PATTERN FAILURES
2657                                                          ;IN MAPPING REGISTERS
2658
2659
2660
2661                          ;;*****************************************************************
2662                          ;*TEST 14       COUNT PATTERN HIGH 20 MAP REGISTERS LOWER 16 BITS
2663                          ;*
2664                          ;*      THIS TEST WILL RUN A COUNT PATTERN THROUGH UNIBUS MAP REGISTERS
2665                          ;*      MAPL20 - MAPL37.  IF THE COUNT RECIEVED DOES NOT MATCH THE
2666                          ;*      EXPECTED PATTERN THEN THE MAP REGISTER ADDRESS, DATA RECEIVED,
2667                          ;*      PATTERN LOADED, AND PATTERN EXPECTED ARE REPORTED.  AT THE END
2668                          ;*      OF THE TEST A SUMMARY OF ALL ERRORS IS GIVEN SO THAT YOU CAN
2669                          ;*      DETERMINE IF THE ERROR IS BIT SENSITIVE OR REGISTER SENSITIVE.
2670                          ;*      THE REGISTERS ARE ON 'MAPD' AND THE OUTPUT MULTIPLEXER IS ON
2671                          ;*      'MAPJ'.
2672                          ;;*****************************************************************
2673  012424                  TST14:
2674  012424  000004                  SCOPE
2675  012426  012737  012550  001324  MOV     #TST15,NXTTST   ;SAVE STARTING ADDRESS OF NEXT TEST
2676                                                          ;FOR ESCAPE ON PARITY ERRORS
2677  012434  012737  000012  001204  MOV     #12.,$TIMES     ;;DO 12 ITERATIONS
2678  012442  005002           20$:   CLR     R2              ;START WITH AN ALL ZERO PATTERN
2679  012444  012737  012464  001110  MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
2680  012452  010204           1$:    MOV     R2,R4           ;PUT COUNT IN R4 FOR MASK
2681  012454  042704  000001          BIC     #000001,R4      ;CLEAR BITS IN MASK FOR PROPER COMPARE
2682  012460  012700  170300          MOV     #MAPL20,R0      ;LOAD STARTING MAP REGISTER ADDRESS
2683  012464  000240           2$:    NOP                     ;THIS IS A SYNC POINT FOR SCOPING
2684  012466  010210                  MOV     R2,(R0)         ;LOAD MAP REGISTER WITH COUNT PATTERN
2685  012470  011003                  MOV     (R0),R3         ;READ MAP REGISTER INTO R3
2686  012472  020403                  CMP     R4,R3           ;COMPARE MASK WITH DATA
2687  012474  001402                  BEQ     3$              ;BRANCH IF DATA MATCHES
2688  012476  004737  005432          JSR     PC,COUNT        ;JUMP TO COUNT TO LOG ALL ERRORS
2689  012502  005003           3$:    CLR     R3              ;CLEAR DATA HOLDER
2690  012504  062700  000004          ADD     #4,R0           ;POINT TO NEXT MAP REGISTER UNDER TEST
2691  012510  022700  170374          CMP     #MAPL37,R0      ;SEE IF ALL REGISTERS HAVE BEEN LOADED
2692  012514  103363                  BHIS    2$              ;BRANCH IF STILL MORE TO GO
```

```
012516 022702 177777          CMP     #177777,R2      ;SEE IF COUNT HAS CYCLED
012522 001403                 BEQ     4$              ;BRANCH IF TEST IS DONE
012524 062702 000401          ADD     #401,R2         ;CHANGE COUNT PATTERN
012530 000750                 BR      1$              ;CONTINUE TESTING
012532 012737 012442 001110 4$:MOV    #20$,$LPERR     ;INITIALIZE LOOPING POINTER IN CASE
                                                      ;OF INTERMITTENT FAULTS.
012540 005737 001254          TST     ERRCNT          ;SEE IF THERE WERE ANY ERRORS
012544 001401                 BEQ     TST15           ;;BRANCH IF NO ERRORS
012546 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'.
                        ;*****************************************************************
012550                  TST15:
                                SCOPE
012550 000004
012552 012737 012674 001324     MOV     #TST16,NXTTST   ;SAVE STARTING ADDRESS OF NEXT TEST
                                                        ;FOR ESCAPE ON PARITY ERRORS
012560 012737 000144 001204     MOV     #144,$TIMES     ;;DO 144 ITERATIONS
012566 005002                   CLR     R2              ;START WITH AN ALL ZERO PATTERN
012570 012737 012610 001110 20$:MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
012576 010204               1$: MOV     R2,R4           ;PUT COUNT IN R4 FOR MASK
012600 042704 177700            BIC     #177700,R4      ;CLEAR BITS IN MASK FOR PROPER COMPARE
012604 012700 170302            MOV     #MAPH20,R0      ;LOAD STARTING MAP REGISTER ADDRESS
012610 000240               2$: NOP                     ;THIS IS A SYNC POINT FOR SCOPING
012612 010210                   MOV     R2,(R0)         ;LOAD MAP REGISTER WITH COUNT PATTERN
012614 011003                   MOV     (R0),R3         ;READ MAP REGISTER INTO R3
012616 020403                   CMP     R4,R3           ;COMPARE MASK WITH DATA
012620 001402                   BEQ     3$              ;BRANCH IF DATA MATCHES
012622 004737 005432            JSR     PC,COUNT        ;JUMP TO COUNT TO LOG ALL ERRORS
012626 005003               3$: CLR     R3              ;CLEAR DATA HOLDER
012630 062700 000004            ADD     #4,R0           ;POINT TO NEXT MAP REGISTER UNDER TEST
012634 022700 170376            CMP     #MAPH37,R0      ;SEE IF ALL REGISTERS HAVE BEEN LOADED
012640 103363                   BHIS    2$              ;BRANCH IF STILL MORE TO GO
012642 022702 000177            CMP     #177,R2 ;SEE IF COUNT HAS CYCLED
012646 001403                   BEQ     4$              ;BRANCH IF TEST IS DONE
012650 062702 000001            ADD     #1,R2           ;CHANGE COUNT PATTERN
012654 000750                   BR      1$              ;CONTINUE TESTING
012656 012737 012566 001110 4$:MOV     #20$,$LPERR     ;INITIALIZE LOOPING POINTER IN CASE
                                                        ;OF INTERMITTENT FAULTS.
012664 005737 001254            TST     ERRCNT          ;SEE IF THERE WERE ANY ERRORS
012670 001401                   BEQ     TST16           ;;BRANCH IF NO ERRORS
012672 104016                   ERROR   16              ;SUMMARY OF COUNT PATTERN FAILURES
                                                        ;IN MAPPING REGISTERS
```

```
                                        ;*****************************************************************
                                        ;*TEST 16       CACHE REGISTER DATA PATHS
                                        ;*
                                        ;*      THIS TEST MAKES AN EFFORT TO VERIFY THE DATA PATH FROM THE
                                        ;*      CACHE REGISTERS THROUGH THE UNIBUS MAP TO THE CPU.  IT FIRST
                                        ;*      CHECKS THE CONTROL REGISTER (177746) TO SEE IF THE CACHE IS
                                        ;*      DISABLED, IF IT IS NOT DISABLED THEN THE CONTROL REGISTER
                                        ;*      IS CLEARED AND VERIFIED TO BE ZERO.  THE MAINTENENCE REGISTER
                                        ;*      IS ALWAYS CLEARED AND VERIFIED TO BE ZERO.  THEN THE CACHE
                                        ;*      ADDRESS REGISTERS (177740 & 177742) ARE READ, WITH THE ERROR
                                        ;*      REGISTER CLEARED,  THEY SHOULD BE 177740 AND 000003 RESPECTIVELY.
                                        ;*      IF ANY OF THESE CONDITIONS ARE NOT MET AN ERROR IS REPORTED
                                        ;*      AS BEING A POSSIBLE CACHE REGISTER DATA PATH ERROR.
                                        ;*      THE INPUT TO THE CACHE REGISTERS IS FROM 'MAPA' AND THE CACHE
                                        ;*      DATA MULTIPLEXER IS ON 'MAPH'.  THE REGISTER INPUTS ARE THE
                                        ;*      'X'' INPPUTS, THESE GO TO 'MAPJ' TO GET BACK TO THE CPU.
                                        ;*
                                        ;*****************************************************************
012674                                  TST16:
                                                SCOPE
012674  000004
012676  012737  013072  001324                MOV     #TST17,NXTTST   ;SAVE STARTING ADDRESS OF NEXT TEST
                                                                        ;FOR ESCAPE ON PARITY ERRORS
012704  012737  012724  001110  20$:     MOV     #10$,$LPERR     ;SET LOOP ON ERROR POINTER TO 10$
012712  005000                          CLR     R0              ;CLEAR R0 IN CASE CACHE IS DISABLED
012714  032737  000014  177746          BIT     #14,@#CONTRL    ;SEE IF EITHER GROUP IS DISABLED
012722  001012                          BNE     1$              ;BRANCH IF CACHE IS DISABLED
012724  000240                  10$:    NOP                     ;THIS IS A SYNC POINT FOR SCOPING
012726  005005                          CLR     R5
012730  005037  177746                  CLR     @#CONTRL        ;CLEAR CONTROL REGISTER
012734  013700  177746          21$:    MOV     @#CONTRL,R0     ;READ CONTROL REGISTER
012740  001403                          BEQ     1$              ;BRANCH IF CONTROL REGISTER IS ZERO
012742  005205                          INC     R5              ;WAIT FOR VCIP BIT IN CCR TO
012744  001373                          BNE     21$             ;CLEAR
012746  104017                          ERROR   17              ;POSSIBLE FAULT IN CACHE REG DATA PATHS
012750  012737  012756  001110  1$:     MOV     #11$,$LPERR     ;SET LOOP ON ERROR POINTER TO 11$
012756  000240                  11$:    NOP                     ;THIS IS A SYNC POINT FOR SCOPING
012760  005037  177750                  CLR     @#MAINT         ;CLEAR MAINTENENCE REGISTER
012764  013701  177750                  MOV     @#MAINT,R1      ;READ MAINTENENCE REGISTER
012770  001401                          BEQ     2$              ;BRANCH IF MAINTENENCE REGISTER IS ZERO
012772  104020                          ERROR   20              ;MAINTENENCE REGISTERWILL NOT CLEAR
012774  012737  013002  001110  2$:     MOV     #12$,$LPERR     ;SET LOOP ON ERROR POINTER TO 12$
013002  012737  177777  177744  12$:    MOV     #-1,MEMERR      ;MAKE SURE THAT CACHE ERROR REG IS CLEAR
013010  000240                          NOP                     ;THIS IS A SYNC POINT FOR SCOPING
013012  013700  177740                  MOV     LOADRS,R0       ;READ CACHE LOW ADDR REG TO R0
013016  022700  177740                  CMP     #177740,R0      ;SEE IF R0 = ADDR OF CACHE LOW ADDR REG
013022  001401                          BEQ     3$              ;BRANCH IF DATA READ CORRECTLY
013024  104021                          ERROR   21              ;CANNOT READ 177740 FROM 'LOADRS''
013026  012737  013034  001110  3$:     MOV     #13$,$LPERR     ;SET LOOP ON ERROR POINTER TO 13$
013034  012737  177777  177744  13$:    MOV     #-1,MEMERR      ;MAKE SURE THAT CACHE ERROR REG IS CLEAR
013042  000240                          NOP                     ;THIS IS A SYNC POINT FOR SCOPING
013044  013700  177742                  MOV     HIADRS,R0       ;READ CACHE HIGH ADDR REG TO R0
013050  042700  177700                  BIC     #177700,R0      ;MASK OFF UPPER  10 BITS
013054  022700  000003                  CMP     #3,R0           ;SEE IF R0 = 3
013060  001401                          BEQ     14$             ;BRANCH IF DATA MATCHES
013062  104022                          ERROR   22              ;CANNOT READ 000003 FROM 'HIADRS''
013064  012737  012704  001110  14$:    MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
```

```
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826  013072                          ;*****************************************************************
2827  013072                          ;*TEST 17       CACHE REGISTER DATA PATH COUNT PATTERN
2828  013072                          ;*
2829  013072                          ;*      THIS TEST FIRST SAVES THE CONDITION OF THE CACHE CONTROL
2830  013072                          ;*      REGISTER IS 'STMP5'' SO THAT THE CACHE CAN BE RESTORED.
2831  013072                          ;*      IF EITHER BIT 3 OR 4 IS ON THE TEST BRANCHES TO '3$'' TO
2832  013072                          ;*      TEST BITS 5 - 15 OF THE DATA PATH, IF NEITHER BIT IS ON THEN
2833  013072                          ;*      THE TEST RUNS A COUNT PATTERN THRU THE CONTROL REGISTER FROM
2834  013072                          ;*      00 - 77.   IF ANY BITS ARE DETECTED AS BEING STUCK THE ERROR
2835  013072                          ;*      IS REPORTED AND A SUMMARY OF THOSE ERRORS IS GIVEN.  THEN THE
2836  013072                          ;*      TEST FORCES MISSES IN BOTH GROUPS OF THE CACHE AND PROCEEDS
2837  013072                          ;*      TO RUN A COUNT PATTERN THROUGH THE CACHE MAINTENENCE REGISTER
2838  013072                          ;*      (177746) FROM 000000 - 177760 IN INCREMENTS OF 20.
2839  013072                          ;*      THE CODE IS SET UP IN SUCH A WAY THAT NO PARITY ERRORS
2840  013072                          ;*      SHOULD BE GENERATED.  AGAIN ALL BAD MATCHES OF DATA WILL BE
2841  013072                          ;*      REPORTED AND A SUMMARY OF ALL ERRORS IS GIVEN AT THE END OF
2842  013072                          ;*      THE TEST AND THE CACHE IS RESTORED TO ITS PREVIOUS CONDITION.
2843  013072                          ;*
2844  013072                          ;*****************************************************************
2845  013072                  TST17:
2846  013072  000004                         SCOPE
2847  013074  012737  013514  001324         MOV     #TST20,NXTTST   ;SAVE STARTING ADDRESS OF NEXT TEST
2848                                                                  ;FOR ESCAPE ON PARITY ERRORS
2849  013102  012737  000012  001204         MOV     #12,$TIMES      ;;DO 12 ITERATIONS
2850  013110  104416                         TBITO                   ;TURN OFF T BIT TRAPPING IF ON
2851  013112  013737  177746  001202  20$:   MOV     @#CNTRL,$TMP5   ;SAVE CONTROL REG TO RESTORE CACHE
2852  013120  032737  000014  177746         BIT     #14,@#CNTRL     ;SEE IF EITHER GROUP IS DISABLED
2853  013126  001047                         BNE     3$              ;SKIP FIRST PART OF TEST IF DISABLED
2854  013130  012700  177746                 MOV     #CNTRL,R0       ;LOAD ADDR OF CONTROL REG INTO R0
2855  013134  005002                         CLR     R2              ;START COUNT PATTERN AT ZERO
2856  013136  012737  013144  001110         MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
2857  013144  000240                 1$:     NOP                     ;THIS IS A SYNC POINT FOR SCOPING
2858  013146  010210                         MOV     R2,(R0)         ;LOAD COUNT INTO CONTROL REG
2859  013150  011003                         MOV     (R0),R3         ;READ COUNT FROM CONTROL REG
2860  013152  020203                         CMP     R2,R3           ;SEE IF COUNT = DATA READ
2861  013154  001402                         BEQ     2$              ;BRANCH IF DATA MATCHES
2862  013156  004737  005576                 JSR     PC,CACOUNT      ;LOG AND REPORT COUNT PATTERN ERROR
2863  013162  005202                 2$:     INC     R2              ;CHANGE COUNT PATTERN
2864  013164  022702  000077                 CMP     #77,R2          ;SEE IF COUNT HAS PASSED MAXIMUM
2865  013170  103365                         BHIS    1$              ;BRANCH IF COUNT HASN'T CYCLED
2866  013172  005737  001254                 TST     ERRCNT          ;SEE IF ANY ERRORS ON THIS CYCLE
2867  013176  001423                         BEQ     3$              ;BRANCH IF NO ERRORS
2868  013200  104023                         ERROR   23              ;COUNT THRU CONTROL REG FAILED
2869  013202  005037  001322                 CLR     RETRY           ;CLEAR RETRY FLAG AN THE START OF
2870                                                                  ;EACH TEST
2871  013206  005037  001254                 CLR     ERRCNT          ;CLEAR THE MULTIPLE ERROR COUNTER
2872  013212  005037  001232                 CLR     DATAOR          ;LOCATION FOR LOGICAL OR OF BAD DATA
2873  013216  005037  001226                 CLR     ADDROR          ;LOCATION FOR LOGICAL OR OF ADDRESS
2874  013222  005037  001236                 CLR     PATTOR          ;LOCATION FOR LOGICAL OR OF PATTERN LOADED
2875  013226  012700  177777                 MOV     #-1,R0          ;LOAD -1 INTO R0 TO INITIALIZE LOGICAL AND LOCS
2876  013232  010037  001230                 MOV     R0,DATAND       ;LOCATION FOR LOGICAL AND OF BAD DATA
2877  013236  010037  001224                 MOV     R0,ADRAND       ;LOCATION FOR LOGICAL AND OF ADDRESS
2878  013242  010037  001234                 MOV     R0,PATAND       ;LOCATION FOR LOGICAL AND OF PATTERN LOADED
2879  013246  012737  000014  177746  3$:    MOV     #14,@#CNTRL     ;DISABLE BOTH GROUPS OF CACHE
```

```
2861  013254  012700  177750            MOV     #MAINT,R0       ;LOAD ADDR OF MAINTENENCE REG INTO R0
2862  013260  010001                    MOV     R0,R1           ;R1 HAS #MAINT SO CLR (R1) HAS P.B.'S ON
2863  013262  005002                    CLR     R2              ;START COUNT PATTERN AT ZERO
2864  013264  012737  013272  001110    MOV     #4$,$LPERR      ;SET LOOP ON ERROR POINTER TO 4$
2865  013272  010205            4$:     MOV     R2,R5           ;PUT COUNT IN R5 FOR CORRECT PARITY
2866  013274  000240                    NOP                     ;THIS IS A SYNC POINT FOR SCOPING
2867  013276  010510                    MOV     R5,(R0)         ;LOAD MAINT REGISTER INST HAS P.B.'S ON
2868  013300  011003                    MOV     (R0),R3         ;READ MAINT REG (P.B.'S ON)
2869  013302  005011                    CLR     (R1)            ;TURN OFF MAINT REG (P.B.'S ON)
2870  013304  050000                    BIS     R0,R0           ;DUMMY INST WITH P.B.'S ON
2871  013306  020203                    CMP     R2,R3           ;SEE IF COUNT = DATA READ
2872  013310  001402                    BEQ     5$              ;BRANCH IF DATA MATCHES
2873  013312  004737  005576            JSR     PC,CACOUNT      ;LOG AND REPORT COUNT PATTERN ERROR
2874  013316  062702  000020    5$:     ADD     #20,R2          ;CHANGE COUNT PATTERN
2875  013322  001363                    BNE     4$              ;BRANCH IF COUNT HAS NOT PASSED 177760
2876  013324  005737  001254            TST     ERRCNT          ;SEE IF ANY ERRORS ON THIS COUNT CYCLE
2877  013330  001401                    BEQ     19$             ;BRANCH IF NO ERRORS ON THIS PART
2878  013332  104024                    ERROR   24              ;SUMMARY OF COUNT ERRORS IN MAINT REG
2879  013334  012737  013112  001110  19$:     MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
2880  013342  013737  001202  177746          MOV     $TMP5,@#CONTRL  ;RETURN CACHE TO PREVIOUS STATE
2881  013350  000004                    SCOPE                   ;
2882
2883                                    ;*      AT THIS POINT 22-BIT RELOCATION FROM MEMORY MANAGEMENT
2884                                    ;*      IS ENABLED, WITH THE KIPAR'S MAPPED TO PHYSICAL 0-24K.
2885                                    ;*      KIPAR6 IS MAPPED TO THE UNIBUS (170000) AND
2886                                    ;*      KIPAR7 IS MAPPED TO THE I/O PAGE (177600).
2887
2888  013352  104420                    TBITR                   ;RESTORE THE T BIT TO ITS CONDITION
                                                                  ;BEFORE THE LAST TEST
2889
2890  013354  012700  077406            MOV     #77406,R0       ;MAKE THE KERNEL I-SPACE PAGES ALL
2891                                                              ;4K, UPWARD EXPANDABLE, READ/WRITE
2892  013360  010037  172300            MOV     R0,KIPDR0       ;KERNEL I-SPACE PAGE 0
2893  013364  010037  172302            MOV     R0,KIPDR1       ;KERNEL I-SPACE PAGE 1
2894  013370  010037  172304            MOV     R0,KIPDR2       ;KERNEL I-SPACE PAGE 2
2895  013374  010037  172306            MOV     R0,KIPDR3       ;KERNEL I-SPACE PAGE 3
2896  013400  010037  172310            MOV     R0,KIPDR4       ;KERNEL I-SPACE PAGE 4
2897  013404  010037  172312            MOV     R0,KIPDR5       ;KERNEL I-SPACE PAGE 5
2898  013410  010037  172314            MOV     R0,KIPDR6       ;KERNEL I-SPACE PAGE 6
2899  013414  010037  172316            MOV     R0,KIPDR7       ;KERNEL I-SPACE PAGE 7
2900  013420  012737  000000  172340    MOV     #000,KIPAR0     ;MAP TO PHYSICAL 0
2901  013426  012737  000200  172342    MOV     #200,KIPAR1     ;MAP TO PHYSICAL 4K - 8K
2902  013434  012737  000400  172344    MOV     #400,KIPAR2     ;MAP TO PHYSICAL 8K - 12K
2903  013442  012737  000600  172346    MOV     #600,KIPAR3     ;MAP TO PHYSICAL 12K - 16K
2904  013450  012737  001000  172350    MOV     #1000,KIPAR4    ;MAP TO PHYSICAL 16K - 20K
2905  013456  012737  001200  172352    MOV     #1200,KIPAR5    ;MAP TO PHYSICAL 20K - 24K
2906  013464  012737  170000  172354    MOV     #170000,KIPAR6  ;MAP TO UNIBUS
2907  013472  012737  177600  172356    MOV     #177600,KIPAR7  ;MAP TO I/O PAGE
2908  013500  012737  000001  177572    MOV     #BIT0,MMR0      ;ENABLE FULL 18-BIT MAPPING
2909  013506  012737  000020  172516    MOV     #BIT4,MMR3      ;ENABLE 22-BIT MAPPING
2910
2911                            ;;**************************************************************
2912                            ;*TEST 20       MAP REGISTER ADDRESS DECODE TEST
2913                            ;*
2914                            ;*      THIS TEST TRIES TO VERIFY THAT NONE OF THE INPUTS TO THE ADDRESS
2915                            ;*      DECODER FOR THE UNIBUS MAP REGISTERS ON 'MAPB' IS STUCK TRUE.
2916                            ;*      KIPAR6 IS SETUP TO HOLD 177702 AND R4 HAS THE VIRTUAL ADDRESS
```

```
                                        ;*      TO SELECT MAPL00, THRU KIPAR6.  THE TEST THEN CHANGES ONE BIT
                                        ;*      AT A TIME IN PAR6 SO THAT IT SHOULD NEVER REFERENCE MAPL00.  IF
                                        ;*      IT DOES AN ERROR IS REPORTED.
                                        ;*
                                        ;**************************************************************************
013514                                  TST20:
                                                SCOPE
013514  000004
013516  012737  013546  001106                 MOV     #20$,$LPADR     ;SET LOOP ON TEST POINTER TO 20$
013524  012737  013546  001110                 MOV     #20$,$LPERR     ;SET LOOP ON ERROR POINTER TO 20$
013532  012737  000020  001102                 MOV     #20,$TSTNM      ;SETUP TEST NUMBER AND CLR ERROR FLAG
013540  013737  001102  177570                 MOV     $TSTNM,DISPLAY  ;DISPLAY TEST NUMBER FOR ALL TO SEE
                                                .EQUIV  BIT4,TIMOUT     ;BIT4 IS TIMEOUT BIT IN CPU ERROR REG
013546  012737  000020  001264  20$:           MOV     #TIMOUT,CPUEXP  ;EXPECTING CPU TIME OUT ON UNIBUS
013554  012737  013614  001110                 MOV     #10$,$LPERR     ;SET LOOP ON ERROR POINTER TO 10$
013562  012737  177702  172354                 MOV     #177702,KIPAR6  ;PUT MAP REG 0 ADDR IN PAR6
013570  012702  175254                        MOV     #175254,R2      ;PATTERN FOR TESTING.
013574  010237  170200                        MOV     R2,@#MAPL0      ;LOAD MAP REGISTER 0
013600  012700  004000                        MOV     #BIT11,R0       ;SET BIT 11 TO FLOAT THRU PAR6
013604  012704  140000                        MOV     #140000,R4      ;VIRT.ADDR. TO SELECT PAR6
013610  074037  172354          1$:           XOR     R0,KIPAR6       ;CHANGE A BIT OF MAP REG 0'S ADDR
013614  000240                  10$:          NOP                     ;THIS IS A SYNC POINT FOR SCOPING
013616  011401                                MOV     (R4),R1         ;READ LOCATION POINTED TO BY PAR6
013620  020201                                CMP     R2,R1           ;SEE IF DATA FETCHED MATCHES PATTERN
013622  001007                                BNE     3$              ;BRANCH IF NOT SAME
013624  012714  000000                        MOV     #0,(R4)         ;TRY TO CLEAR THIS LOCATION
013630  005737  170200                        TST     @#MAPL0         ;SEE IF MAP REG 0 GOT CLEARED
013634  001001                                BNE     2$              ;BRANCH IF MAP REG NOT ZERO
013636  104025                                ERROR   25              ;GOT TO MAPL0 WITH ONE BIT
                                                                        ;DIFFERENT IN ADDRESS FROM 770200
013640  010114                  2$:           MOV     R1,(R4)         ;RELOAD THE LOCATION
013642  074037  172354          3$:           XOR     R0,KIPAR6       ;RESTORE BIT TO ORIGINAL STATUS
013646  006200                                ASR     R0              ;RIGHT SHIFT ONE PLACE
013650  020027  000001                        CMP     R0,#1           ;SEE IF TEST IS OVER
013654  001355                                BNE     1$              ;BRANCH TO CONTINUE TEST
013656  012737  013546  001110                 MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
013664  005037  001264                        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.
                                        ;*
                                        ;**************************************************************************
013670                                  TST21:
                                                SCOPE
013670  000004
013672  012737  014072  001324                 MOV     #TST22,NXTTST   ;SAVE STARTING ADDRESS OF NEXT TEST
                                                                        ;FOR ESCAPE ON PARITY ERRORS
013700  012737  000020  001264  20$:           MOV     #TIMOUT,CPUEXP  ;EXPECTING UNIBUS TIMEOUT IN THIS TEST
013706  012737  013742  001110                 MOV     #10$,$LPERR     ;SET LOOP ON ERROR POINTER TO 10$
013714  012737  177777  172354                 MOV     #177777,KIPAR6  ;PUT CACHE REG BASE IN PAR6
013722  012702  177740                        MOV     #177740,R2      ;PUT DATA PATTERN IN R2
```

```
2973  013726  012700  000040            MOV     #BIT5,R0        ;SET BIT 5 TO FLOAT THRU PAR6
2974  013732  012704  140040            MOV     #140040,R4      ;VIRT. ADDR. TO SELECT PAR6
2975  013736  074037  172354    1$:     XOR     R0,KIPAR6       ;CHANGE A BIT OF CACHE REG'S ADDR
2976  013742  000240            10$:    NOP                     ;THIS IS A SYNC POINT FOR SCOPING
2977  013744  011401                    MOV     (R4),R1         ;TRY TO READ CACHE 'LOADRS' WITH
2978                                                            ;ONE BIT DIFFERENT THAN ADDR. 777740
2979  013746  020201                    CMP     R2,R1           ;SEE IF YOU GOT CACHE 'LOADRS'
2980  013750  001001                    BNE     2$              ;BRANCH TO CONTINUE TEST.
2981  013752  104026                    ERROR   26              ;READ CACHE 'LOADRS' WITHOUT ADRS 777740
2982  013754  074037  172354    2$:     XOR     R0,KIPAR6       ;RESTORE BIT TO ORIGINAL STATUS.
2983  013760  006200                    ASR     R0              ;RIGHT SHIFT R0 1 PLACE
2984  013762  001365                    BNE     1$              ;BRANCH IF NOT 0
2985  013764  012737  013776  001110    MOV     #11$,$LPERR     ;SET LOOP ON ERROR POINTER TO 11$
2986  013772  012701  177700            MOV     #177700,R1      ;PUT ADDRESS IN R1
2987  013776  000240            11$:    NOP                     ;THIS IS A SYNC POINT FOR SCOPING
2988  014000  011103                    MOV     (R1),R3         ;TRY TO READ 177700
2989  014002  020203                    CMP     R2,R3           ;SEE IF YOU GET 177740
2990  014004  001001                    BNE     3$              ;BRANCH IF YOU DON'T
2991  014006  104027                    ERROR   27              ;READ CACHE 'LOADRS' WITHOUT ADRS 777740
2992  014010  012737  014022  001110    MOV     #12$,$LPERR     ;SET LOOP ON ERROR POINTER TO 12$
2993  014016  012701  177760            MOV     #177760,R1      ;PUT ADDRESS IN R1
2994  014022  000240            12$:    NOP                     ;THIS IS A SYNC POINT FOR SCOPING
2995  014024  011103                    MOV     (R1),R3         ;TRY TO READ 177760
2996  014026  020203                    CMP     R2,R3           ;SEE IF YOU GET 177740
2997  014030  001001                    BNE     4$              ;BRANCH IF YOU DON'T
2998  014032  104027                    ERROR   27              ;READ CACHE 'LOADRS' WITHOUT ADRS 777740
2999  014034  012737  014046  001110    MOV     #13$,$LPERR     ;SET LOOP ON ERROR POINTER TO 13$
3000  014042  012701  177754            MOV     #177754,R1      ;PUT ADDRESS IN R1
3001  014046  000240            13$:    NOP                     ;THIS IS A SYNC POINT FOR SCOPING
3002  014050  011103                    MOV     (R1),R3         ;TRY TO READ 177754
3003  014052  020203                    CMP     R2,R3           ;SEE IF YOU GET 177740
3004  014054  001001                    BNE     19$             ;BRANCH IF YOU DIDN'T READ ADDRS 177740
3005  014056  104027                    ERROR   27              ;READ CACHE 'LOADRS' WITHOUT ADRS 777740
3006  014060  012737  013700  001110    MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
3007  014066  005037  001264            CLR     CPUEXP          ;ZERO EXPECTED CPU TRAP CONDITION
3008
3009                                    ;*****************************************************************
3010                                    ;*TEST 22       DATA PATH, UNIBUS TO MAIN MEMORY
3011                                    ;*
3012                                    ;*      THIS TEST RUNS A COUNT PATTERN THROUGH A MEMORY LOCATION VIA
3013                                    ;*      THE UNIBUS.  THE UNIBUS MAP IS LEFT OFF DURING THIS TEST SO
3014                                    ;*      THAT THE ADDRESS IS NOT RELOCATED.  THE TEST TRIES TO LOAD THE
3015                                    ;*      PATTERN INTO ADDRESS 040000 (8K) BUT IF THE MAP JUMPERS ARE
3016                                    ;*      SET NOT TO RESPOND TO THAT ADDRESS THE NEXT 4K IS TRIED UNTIL
3017                                    ;*      THE TEST GETS TO MAIN MEMORY FROM THE UNIBUS.  IF THIS TEST
3018                                    ;*      DETERMINES THAT IT CANNOT GET TO MAIN MEMORY FROM THE UNIBUS
3019                                    ;*      IT REPORTS THE FACT AND SKIPS THE NEXT TEST FOR VERIFICATION.
3020                                    ;*      THE DATA PATH TO MAIN MEMORY FROM THE UNIBUS IS ON 'MAPA' AND
3021                                    ;*      THE PATH FROM MAIN MEMORY TO THE UNIBUS IS FROM THE CACHE
3022                                    ;*      'DTML CDMX D00' - 'DTML CDMX D15' INTO 'MAPH' THE 'Z' INPUTS
3023                                    ;*      TO THE MULTIPLEXER AND THEN TO 'MAPJ' AND OUT TO THE UNIBUS.
3024                                    ;*
3025                                    ;*****************************************************************
3026
3027  014072                    TST22:
3028  014072  000004                    SCOPE
```

```
3029  014074  012737  014324  001324          MOV     #TST23,NXTTST   ;SAVE STARTING ADDRESS OF NEXT TEST
3030                                                                  ;FOR ESCAPE ON PARITY ERRORS
3031  014102  012737  000012  001204          MOV     #12.,$TIMES     ;;DO 12 ITERATIONS
3032  014110  004737  005134              20$: JSR     PC,CLRMAP       ;CLEAR ALL MAP REGISTERS
3033  014114  012737  000020  001264          MOV     #TIMOUT,CPUEXP  ;TIMEOUTS MIGHT OCCUR IN THIS TEST.
3034  014122  012737  014130  001110          MOV     #10$,$LPERR     ;SET LOOP ON ERROR POINTER TO 10$
3035  014130  012737  170400  172354  10$:    MOV     #170400,KIPAR6  ;START WITH ADDRESS 8K FROM UNIBUS
3036  014136  005037  001266          1$:     CLR     PCPUER          ;CLEAR ERROR CONDITION LOCATION
3037  014142  000240                          NOP                     ;THIS IS A SYNC POINT FOR SCOPING
3038  014144  013700  140000                  MOV     @#140000,R0     ;TRY TO READ ADDRESS POINTED TO BY PAR6
3039  014150  005737  001266                  TST     PCPUER          ;SEE IF READ OF ADDRESS TIMED OUT
3040  014154  001412                          BEQ     2$              ;BRANCH IF REFERENCE WAS GOOD
3041  014156  062737  000200  172354  4$:     ADD     #200,KIPAR6     ;TRY NEXT 4K BLOCK OF MEMORY
3042  014164  022737  177600  172354          CMP     #177600,KIPAR6  ;SEE IF YOU'VE POINTED TO I/O PAGE
3043  014172  001361                          BNE     1$              ;BRANCH IF NOT
3044  014174  104030                          ERROR   30              ;NO UNIBUS ADDRESSES RESPOND
3045  014176  000137  014454                  JMP     SIZEJ           ;JUMP TO SIZE JUMPER TEST
3046  014202  013737  172354  172352  2$:     MOV     KIPAR6,KIPAR5   ;PUT PAR6 INTO PAR5
3047  014210  042737  170000  172352          BIC     #170000,KIPAR5  ;MAKE PAR5 A NON UNIBUS ADDRESS
3048  014216  012737  173214  120000          MOV     #173214,@#120000 ;PUT RANDOM NUMBER INTO TEST LOCATION BY FAST BUS
3049  014224  013701  140000                  MOV     @#140000,R1     ;READ TEST LOCATION VIA UNIBUS
3050  014230  022701  173214                  CMP     #173214,R1      ;SEE IF DATA WAS READ PROPERLY
3051  014234  001403                          BEQ     3$              ;DATA OKAY NOW VERIFY DATA PATH
3052  014236  020001                          CMP     R0,R1           ;SEE IF DATA CHANGED FROM FIRST READ
3053  014240  001001                          BNE     3$              ;BRANCH IF DATA CHANGED
3054  014242  000745                          BR      4$              ;TRY NEXT 4K BLOCK
3055  014244  005001              3$:         CLR     R1              ;CLEAR REGISTER TO HOLD COUNT
3056  014246  012737  014260  001110          MOV     #11$,$LPERR     ;SET LOOP ON ERROR POINTER TO 11$
3057  014254  012702  140000                  MOV     #140000,R2      ;LOAD VIRTUAL ADDRESS INTO.R2
3058  014260  000240              11$:        NOP                     ;THIS IS A SYNC POINT FOR SCOPING
3059  014262  010112              5$:         MOV     R1,(R2)         ;LOAD COUNT INTO TEST LOCATION VIA U.B.
3060  014264  011200                          MOV     (R2),R0         ;READ TEST LOCATION BACK VIA UNIBUS
3061  014266  020100                          CMP     R1,R0           ;COMPARE COUNT WITH DATA READ
3062  014270  001402                          BEQ     6$              ;BRANCH IF DATA MATCHES
3063  014272  004737  005522                  JSR     PC,UBCOUNT      ;COUNT FAILED REPORT ERROR
3064  014276  005201              6$:         INC     R1              ;INCREASE COUNT
3065  014300  001370                          BNE     5$              ;BRANCH IF COUNT HASN'T CYCLED
3066  014302  005737  001254                  TST     ERRCNT          ;WERE THERE ANY ERRORS ON THIS TEST
3067  014306  001401                          BEQ     19$             ;BRANCH IF NO ERRORS ON THIS TEST
3068  014310  104031                          ERROR   31              ;SUMMARY OF ERRORS ON THE UNIBUS
3069                                                                  ;DATA PATH
3070  014312  012737  014110  001110  19$:    MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
3071  014320  005037  001264                  CLR     CPUEXP          ;ZERO EXPECTED CPU TRAP CONDITION
3072
3073                          ;;*****************************************************************
3074                          ;;*TEST 23       MAP DOESN'T RELOCATE IF NOT ENABLED
3075                          ;;*
3076                          ;;*     THIS TEST VERIFIES THAT THE UNIBUS MAP DOES NOT RELOCATE IF BITS
3077                          ;;*     OF MMR3 IS NOT SET.  THE TEST ASSUMES THAT THE PREVIOUS TEST HAS
3078                          ;;*     RUN SUCCESSFULLY AND LEFT KIPAR6 POINTING TO THE FIRST UNIBUS
3079                          ;;*     MAPPING REGISTER THAT THE UNIBUS MAP WILL RESPOND TO GREATER
3080                          ;;*     THAN OR EQUAL TO MAPREG #2.  KIPAR5 IS ALSO POINTING TO THE
3081                          ;;*     SAME MEMORY BASE ADDRESS EXCEPT IT POINTS OVER THE FASTBUS.
3082                          ;;*     THE TEST THEN SETS ONE BIT IN EACH A.L.U. OF THE UNIBUS MAP
3083                          ;;*     AND TRIES TO REFERENCE MAIN MEMORY OVER THE UNIBUS.  SINCE THE
3084
```

```
                                        ;*      MAP IS NOT ENABLED THE LOAD WILL GO TO MAIN MEMORY UNRELOCATED.
                                        ;*
                                        ;**************************************************************
014324                                  TST23:
014324  000004                                  SCOPE
014326  012737  014436  001324                   MOV     #TST24,NXTTST   ;SAVE STARTING ADDRESS OF NEXT TEST
                                                                        ;FOR ESCAPE ON PARITY ERRORS
014334  012737  000020  001264          20$:    MOV     #TIMOUT,CPUEXP  ;MIGHT TIME OUT OVER UNIBUS
014342  012737  014400  001110                   MOV     #10$,$LPERR     ;SET LOOP ON ERROR POINTER TO 10$
014350  013700  172354                          MOV     KIPAR6,R0       ;PUT UNIBUS ADDRESS OF MAP REG IN R0
014354  072027  177773                          ASH     #-5,R0          ;RIGHT SHIFT R0 5 PLACES
014360  042700  007400                          BIC     #007400,R0      ;STRIP OFF EXTRANEOUS BITS.
014364  012720  021042                          MOV     #021042,(R0)+   ;SET BOTTOM BIT IN EACH ALU
014370  012710  000042                          MOV     #42,(R0)        ;SET BOTTOM BIT IN EACH ALU
014374  005037  120000                          CLR     @#120000        ;CLEAR TEST LOCATION VIA FAST BUS
014400  000240                          10$:    NOP                     ;THIS IS A SYNC POINT FOR SCOPING
014402  012737  043207  140000                   MOV     #43207,@#140000 ;LOAD TEST LOCATION VIA UNIBUS
                                                                        ;THIS LOAD SHOULD NOT BE RELOCATED
                                                                        ;BY THE UNIBUS MAP, SINCE BIT05 OF
                                                                        ;MMR3 IS CLEAR.
014410  013703  120000                          MOV     @#120000,R3     ;READ TEST LOCATION VIA FAST BUS
014414  022703  043207                          CMP     #43207,R3       ;SEE IF DATA MATCHES
014420  001401                                  BEQ     1$              ;BRANCH IF DATA GOOD
014422  104032                                  ERROR   32              ;MAP RELOCATED WHEN NOT ENABLED
014424  012737  014334  001110          1$:     MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
014432  005037  001264                          CLR     CPUEXP          ;ZERO EXPECTED CPU TRAP CONDITION

                                        ;**************************************************************
                                        ;*TEST 24       SIZE JUMPER LOCATION
                                        ;*
                                        ;*      THIS TEST DETERMINES THE SETTING OF THE JUMPERS ON THE UNIBUS
                                        ;*      MAP WHICH ALLOW THE MAP TO RESPOND TO THOSE ADDRESSES BETWEEN
                                        ;*      THE JUMPER RANGE.  THE DEFAULT SETTING ALLOWS THE MAP TO RESPOND
                                        ;*      TO ADDRESSES 000000 - 757776 ON THE UNIBUS.  IF THE JUMPERS ARE
                                        ;*      NOT SET IN THEIR DEFAULT POSITION AN ERROR MESSAGE IS GIVEN, AND
                                        ;*      THE NUMBER THAT IS REMOVED BY THE JUMPER SETTING IS COMPARED WITH
                                        ;*      THE NUMBER THAT SHOULD NOT RESPOND.  IF THESE NUMBERS DON'T
                                        ;*      CORRESPOND THE ERROR COULD BE IN THE COMPARE CIRCUIT ON 'MAPX'.
                                        ;*
                                        ;**************************************************************
014436                                  TST24:
014436  000004                                  SCOPE
014440  012737  015074  001324                   MOV     #TST25,NXTTST   ;SAVE STARTING ADDRESS OF NEXT TEST
                                                                        ;FOR ESCAPE ON PARITY ERRORS
014446  012737  000001  001204                   MOV     #1,$TIMES       ;;DO 1 ITERATION
014454                                  SIZEJ:
014454  000004                                  SCOPE
014456  012737  014506  001106                   MOV     #20$,$LPADR     ;SET LOOP ON TEST POINTER TO 20$
014464  012737  014506  001110                   MOV     #20$,$LPERR     ;SET LOOP ON ERROR POINTER TO 20$
014472  012737  000024  001102                   MOV     #24,$TSTNM      ;SETUP TEST NUMBER AND CLR ERROR FLAG
014500  013737  001102  177570                   MOV     $TSTNM,DISPLAY  ;DISPLAY TEST NUMBER FOR ALL TO SEE
014506  012737  000020  001264          20$:    MOV     #TIMOUT,CPUEXP  ;EXPECTING CPU TIME OUT ON UNIBUS
014514  012700  170200                          MOV     #MAPLO,R0       ;LOAD ADDRESS OF FIRST MAP REG
014520  012720  020000                  1$:     MOV     #20000,(R0)+    ;LOAD 4K INTO ALL MAP REGISTERS
014524  005020                                  CLR     (R0)+           ;INSURE THAT ALL REGS HAVE UPPER BITS CLR
```

```
3141  014526  022700  170374              CMP     #MAPL37,R0      ;SEE IF LAST REG IS LOADED
3142  014532  101372                      BHI     1$              ;BRANCH IF THERE ARE MORE TO LOAD
3143  014534  052737  000040  172516      BIS     #BIT5,@#MMR3    ;TURN ON MAP RELOCATION
3144  014542  012700  117776              MOV     #117776,R0      ;THIS WILL BE USED TO SELECT PAR 4
3145  014546  012737  170000  172350      MOV     #170000,KIPAR4  ;WE START TESTING WITH MAP 0
3146  014554  012701  000200              MOV     #200,R1         ;CONSTANT USED TO ADD TO PAR 4
3147  014560  012702  125252              MOV     #125252,R2      ;CONSTANT TO LOAD INTO LOCATION 37776
3148  014564  005037  037776      2$:     CLR     @#37776         ;CLEAR TEST LOCATION
3149  014570  010210                      MOV     R2,(R0)         ;TRY TO LOAD TEST CELL THROUGH MAP
3150  014572  023702  037776              CMP     @#37776,R2      ;SEE IF TEST LOCATION WAS LOADED
3151  014576  001411                      BEQ     3$              ;BRANCH IF IT WAS LOADED
3152  014600  060157  172350              ADD     R1,KIPAR4       ;CELL NOT LOADED, TEST NEXT MAP REG
3153  014604  022737  177600  172350      CMP     #177600,KIPAR4  ;SEE IF YOU'RE POINTING TO I/O PAGE
3154  014612  001364                      BNE     2$              ;GO TYPE NEXT MAP REGISTER
3155  014614  104033                      ERROR   33              ;FATAL ERROR RESTARTING PROGRAM
3156  014616  000137  010000              JMP     START           ;RESTART PROGRAM
3157  014622  013737  172350  001240  3$: MOV     KIPAR4,LOWEST   ;FOUND THE LOWEST USABLE MAP REG
3158  014630  013737  172350  001242  4$: MOV     KIPAR4,HIGEST   ;THIS WILL END UP BEING LAST USABLE REG
3159  014636  060137  172350              ADD     R1,KIPAR4       ;TRY NEXT MAP REG TO SEE IF IT RESPONDS
3160  014642  022737  177600  172350      CMP     #177600,KIPAR4  ;SEE IF ALL MAP REGS HAVE BEEN TRIED
3161  014650  001433                      BEQ     7$              ;BRANCH IF ALL ARE DONE
3162  014652  005037  037776              CLR     @#37776         ;CLEAR TEST LOCATION
3163  014656  010210                      MOV     R2,(R0)         ;TRY TO LOAD TEST CELL THROUGH THE MAP
3164  014660  023702  037776              CMP     @#37776,R2      ;SEE IF TEST LOCATION WAS LOADED
3165  014664  001761                      BEQ     4$              ;BRANCH IF IT WAS LOADED
3166  014666  005037  037776      5$:     CLR     @#37776         ;CLEAR TEST LOCATION
3167  014672  005237  001254              INC     ERRCNT          ;COUNT OF REGS AFTER LAST ONE USABLE
3168  014676  010210                      MOV     R2,(R0)         ;TRY TO LOAD TEST CELL THRU MAP
3169  014700  023702  037776              CMP     @#37776,R2      ;SEE IF TEST LOCATION WAS LOADED
3170  014704  001402                      BEQ     6$              ;BRANCH IF LOCATION WAS LOADED
3171  014706  005237  001256              INC     CNTR            ;COUNT OF REGS THAT FAILED TO RESPOND
3172  014712  060137  172350      6$:     ADD     R1,KIPAR4       ;POINT TO NEXT MAP REG UNDER TEST
3173  014716  022737  177600  172350      CMP     #177600,KIPAR4  ;SEE IF TEST IS OVER
3174  014724  001360                      BNE     5$              ;BRANCH IF TEST NOT DONE
3175  014726  023737  001254  001256      CMP     ERRCNT,CNTR     ;SEE IF TRIES = FAILURES
3176  014734  001401                      BEQ     7$              ;BRANCH IF EQUAL
3177  014736  104034                      ERROR   34              ;MAP JUMPER COMPARE CIRCUIT BAD
3178  014740  005037  001254      7$:     CLR     ERRCNT          ;CLEAR TRIES COUNTER
3179  014744  005037  001256              CLR     CNTR            ;CLEAR FAILURE COUNTER
3180  014750  005037  001264              CLR     CPUEXP          ;NO CPU TRAPS EXPECTED IN NEXT TEST
3181  014754  005737  001100              TST     $PASS           ;SEE IF THIS IS FIRST PASS
3182  014760  001045                      BNE     TST25           ;;GO TO NEXT TEST IF NOT THE FIRST PASS
3183  014762  023727  001240  170000      CMP     LOWEST,#170000  ;SEE IF LOWER JUMPER IS DEFAULT
3184  014770  001004                      BNE     8$              ;BRANCH IF NOT.
3185  014772  023727  001242  177400      CMP     HIGEST,#177400  ;SEE IF UPPER JUMPER IS DEFAULT.
3186  015000  001401                      BEQ     9$              ;BRANCH IF JUMPERS DEFAULT.
3187  015002  104035              8$:     ERROR   35              ;MAP SIZE JUMPERS NOT DEFAULT
3188                              ;;
3189                              ;*      SETUP POINTERS TO THE LOWEST AND THE HIGHEST USEABLE
3190                              ;*      MAPPING REGISTERS TO CONTINUE WITH TEST.
3191                              ;;
3192  015004  013700  001242      9$:     MOV     HIGEST,R0
3193  015010  013701  001240              MOV     LOWEST,R1
3194  015014  042700  170000              BIC     #170000,R0
3195  015020  042701  170000              BIC     #170000,R1
3196  015024  072027  177773              ASH     #-5,R0          ;RIGHT SHIFT R0 5 PLACES
```

```
3197  015030  072127  177773                  ASH     #-5,R1          ;RIGHT SHIFT R1 5 PLACES
3198  015034  062701  170200                  ADD     #170200,R1
3199  015040  062700  170200                  ADD     #170200,R0
3200  015044  010137  001244                  MOV     R1,LREGL
3201  015050  062701  000002                  ADD     #2,R1           ;POINT TO UPPER BITS OF MAP REG
3202  015054  010137  001246                  MOV     R1,LREGU
3203  015060  010037  001250                  MOV     R0,HREGL
3204  015064  062700  000002                  ADD     #2,R0           ;POINT TO UPPER BITS OF MAP REG
3205  015070  010037  001252                  MOV     R0,HREGU
3206
3207
3208
3209                                  ;*****************************************************************
3210                                  ;*TEST 25       ENSURE THAT THERE IS NO DUAL MAPPING
3211                                  ;*
3212                                  ;*      THIS TEST VERIFIES THAT THE OTHER INPUT (X) TO THE ADDRESS
3213                                  ;*      MULTIPLEXER ON 'MAPC' IS FUNCTIONING PROPERLY.  IT CLEARS
3214                                  ;*      ALL THE MAP REGISTERS EXCEPT THE ONE UNDER TEST, AND LOADS
3215                                  ;*      THAT ONE WITH 00020000.  THE TEST THEN USES A VIRTUAL ADDRESS
3216                                  ;*      TO SELECT THAT MAP REGISTER AND ADD 17776, SO THAT IT SHOULD
3217                                  ;*      REFERENCE ADDRESS 00037776.  A REFERENCE IS MADE THROUGH EACH
3218                                  ;*      OF THE REGISTERS AND ANY THAT FETCH THE CORRECT DATA ARE CHECKED
3219                                  ;*      TO SEE THAT IT WAS THE MAP REGISTER UNDER TEST.  IF NOT BOTH THE
3220                                  ;*      MAP REGISTER UNDER TEST AND THE DUALED REGISTER ARE REPORTED.
3221                                  ;*
3222                                  ;*****************************************************************
3223  015074                          TST25:
3224  015074  000004                          SCOPE
3225  015076  012737  015270  001324          MOV     #TST26,NXTTST   ;SAVE STARTING ADDRESS OF NEXT TEST
3226                                                                  ;FOR ESCAPE ON PARITY ERRORS
3227  015104  012737  000144  001204          MOV     #144.,$TIMES    ;;DO 144 ITERATIONS
3228  015112  004737  005134          20$:    JSR     PC,CLRMAP       ;CLEAR ALL MAP REGISTERS
3229  015116  005037  001170                  CLR     $TMP0           ;USED AS FLAG IN TEST
3230  015122  012703  100000                  MOV     #100000,R3      ;SELECT P.A.R. 4 OFFSET OF ZERO
3231  015126  012737  015160  001110          MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
3232  015134  013702  001244                  MOV     LREGL,R2        ;PUT ADDRESS OF LOWEST USABLE MAP REG IN R2
3233  015140  013700  001240                  MOV     LOWEST,R0       ;MAP REGISTER UNDER TEST IN R0
3234  015144  013701  001240          1$:     MOV     LOWEST,R1       ;MAP REGISTER USED IN CURRENT REFERENCE
3235  015150  012712  040000                  MOV     #40000,(R2)     ;LOAD MAP REG UNDER TEST WITH 8K BASE
3236  015154  010237  040000                  MOV     R2,@#40000      ;LOAD TEST LOCATION WITH THE ADDRESS
3237                                                                  ;OF THE MAP REGISTER UNDER TEST
3238  015160  010137  172350          2$:     MOV     R1,KIPAR4       ;LOAD PAR 4 WITH NEXT MAP REG UNIBUS ADDR
3239  015164  011304                          MOV     (R3),R4         ;READ THROUGH THE MAP
3240  015166  020402                          CMP     R4,R2           ;SEE IF CORRECT DATA WAS FETCHED
3241  015170  001010                          BNE     4$              ;BRANCH IF NO MATCH
3242  015172  020001                          CMP     R0,R1           ;SEE IF MAP REGS ARE THE SAME
3243  015174  001403                          BEQ     3$              ;BRANCH IF CORRECT MAP REG WAS USED
3244  015176  004737  005356                  JSR     PC,DUALADR      ;LOG AND REPORT ALL ERRORS
3245  015202  000403                          BR      4$              ;SKIP NEXT INSTRUCTION
3246  015204  012737  000001  001170  3$:     MOV     #1,$TMP0        ;SET FLAG WHEN ADDRS MATCH
3247  015212  062701  000200          4$:     ADD     #200,R1         ;TRY NEXT MAP REG
3248  015216  023701  001242                  CMP     HIGEST,R1       ;SEE IF ALL HAVE BEEN TRIED
3249  015222  103356                          BHIS    2$              ;BRANCH IF STILL MORE TO TRY
3250  015224  005737  001170                  TST     $TMP0           ;SEE THAT THERE WAS A SUCCESSFUL MATCH
3251  015230  001001                          BNE     5$              ;BRANCH IF THERE WAS
3252  015232  104036                          ERROR   36              ;DID NOT MATCH MAP REG ADDRESS
```

```
                        T25     ENSURE THAT THERE IS NO DUAL MAPPING

3253  015234  005037  001170   5$:    CLR     $TMP0           ;CLEAR FLAG FOR NEXT REG
3254  015240  005012            CLR     (R2)            ;CLEAR MAP REG JUST TESTED
3255  015242  062702  000004    ADD     #4,R2           ;POINT TO NEXT MAP REG TO LOAD
3256  015246  062700  000200    ADD     #200,R0         ;POINT TO NEXT MAP REG UNDER TEST
3257  015252  023700  001242    CMP     HIGEST,R0       ;SEE IF ALL MAP REGS HAVE BEEN TESTED
3258  015256  103332            BHIS    1$              ;BRANCH IF STILL MORE TO TEST
3259  015260  005737  001254    TST     ERRCNT          ;SEE IF THERE WERE ANY ERRORS
3260  015264  001401            BEQ     TST26           ;;BRANCH TO NEXT TEST IF NO ERRORS
3261  015266  104013            ERROR   13              ;ERROR TYPE OUT ITEM 13
3262
3263
3264
3265
3266                    ;;*****************************************************************
3267                    ;*TEST 26       LOAD LOCATIONS 40000 - 137776 WITH THEIR ADDRESSES
3268                    ;*
3269                    ;*      THIS TEST IS USED TO LOAD MAIN MEMORY FROM ADDRESS 00040000 TO
3270                    ;*      ADDRESS 00137776 WITH ITS OWN ADDRESS.  IT THEN CHECKS THAT
3271                    ;*      MEMORY OVER THE UNIBUS AND LOGS ANG REPORTS ANY ERRORS THAT
3272                    ;*      IT FINDS.
3273                    ;*
3274                    ;*****************************************************************
3275  015270            TST26:
3276  015270  000004            SCOPE
3277  015272  012737  015550  001324    MOV     #TST27,NXTTST   ;SAVE STARTING ADDRESS OF NEXT TEST
3278                                                    ;FOR ESCAPE ON PARITY ERRORS
3279  015300  042737  000040  172516    BIC     #BIT5,MMR3      ;TURN OFF MAP RELOCATION
3280  015306  012737  000400  172350    MOV     #400,KIPAR4     ;MAP PAGE 4 TO 8K
3281  015314  012700  040000            MOV     #40000,R0       ;STARTING ADDRESS FOR DATA PATTERN
3282  015320  012701  100000   1$:      MOV     #100000,R1      ;VIRTUAL ADDRESS
3283  015324  012702  010000            MOV     #^D4096,R2      ;LOAD 4096 LOCATIONS AT A TIME
3284  015330  010021   2$:      MOV     R0,(R1)+        ;LOAD PHY. ADDR. INTO EACH MEMORY LOC.
3285  015332  062700  000002            ADD     #2,R0           ;POINT TO NEXT PHYSICAL ADDRESS
3286  015336  077204            SOB     R2,2$           ;BRANCH IF 4K OF MEMORY NOT LOADED
3287  015340  062737  000200  172350    ADD     #200,KIPAR4     ;POINT TO NEXT 4K BANK OF MEMORY
3288  015346  022737  001400  172350    CMP     #1400,KIPAR4    ;SEE IF 24K IS LOADED
3289  015354  101361            BHI     1$              ;BRANCH IF MORE MEMORY TO LOAD
3290                    ;*
3291                    ;*      MEMORY FROM 8K - 24K IS NOW LOADED WITH ITS OWN ADDRE
3292                    ;*
3293  015356  012737  015364  001106            MOV     #20$,$LPADR     ;SET LOOP ADDRESS TO 20$
3294  015364  012737  015422  001110   20$:     MOV     #4$,$LPERR      ;SET LOOP ON ERROR POINTER TO 4$
3295  015372  022737  171400  172354            CMP     #171400,KIPAR6  ;DID I USE ANY MAP REGISTER
                                                        ;BELOW REGISTER 6 (UB. ADDR 140000)
3296
3297  015400  101463            BLOS    TST27           ;;BRANCH TO NEXT TEST IF NOT
3298  015402  013700  172354            MOV     KIPAR6,R0       ;LOAD PAR6 INTO R0 TO GET
                                                        ;THE STARTING DATA PATTERN
3299                                                    ;R0 NOW HOLDS THE STARTING DATA PATTERN
3300  015406  072027  000006            ASH     #6,R0
3301  015412  012701  140000   3$:      MOV     #140000,R1      ;STARTING VIRTUAL ADDRESS
3302  015416  012702  010000            MOV     #^D4096,R2      ;PREPARE TO READ 4K AT A TIME
3303  015422  000240   4$:      NOP                     ;THIS IS A SYNC POINT FOR SCOPING
3304  015424  011103            MOV     (R1),R3         ;READ MAIN MEMORY THRU UNIBUS
3305  015426  020003            CMP     R0,R3           ;SEE IF THE ADDRESSES MATCH
3306  015430  001015            BNE     6$              ;BRANCH IF ERROR
3307  015432  062701  000002   5$:      ADD     #2,R1           ;CHANGE VIRTUAL ADDRESS
3308  015436  062700  000002            ADD     #2,R0           ;CHANGE PHYSICAL ADDRESS
```

```
3309  015442  077211                          SOB     R2,4$           ;BRANCH IF 4K OF MEMORY NOT READ
3310  015444  062737  000200  172354          ADD     #200,KIPAR6     ;POINT TO NEXT BANK OF 4K THRU UNIBUS
3311  015452  022737  171400  172354          CMP     #171400,KIPAR6  ;SEE IF THIS POINTS TO 24K PLUS 2
3312  015460  101354                          BHI     3$              ;BRANCH IF 24K OF MEMORY NOT CHECKED
3313  015462  000423                          BR      10$             ;TEST FINISHED, BRANCH TO EXIT
3314
3315  015464                          6$:
3316  015464  005100                          COM     R0              ;GET R0 READY FOR AND
3317  015466  040037  001224                  BIC     R0,ADRAND               ;PERFORM LOGICAL AND
3318  015472  005100                          COM     R0              ;PUT R0 BACK AS IT WAS
3319  015474  005103                          COM     R3              ;GET R3 READY FOR AND
3320  015476  040337  001230                  BIC     R3,DATAND               ;PERFORM LOGICAL AND
3321  015502  005103                          COM     R3              ;PUT R3 BACK AS IT WAS
3322  015504  050037  001226                  BIS     R0,ADDROR       ;LOGICAL OR OF PHYSICAL ADDRESS
3323  015510  050337  001232                  BIS     R3,DATAOR       ;LOGICAL OR OF ADDRESS FETCHED
3324  015514  005737  001254                  TST     ERRCNT          ;IS THIS THE FIRST ERROR HERE?
3325  015520  001002                          BNE     7$              ;BRANCH IF NOT FIRST ERROR
3326  015522  104206                          ERROR   206             ;DIDN'T READ ADDRESSES RIGHT FROM UNIBUS
3327  015524  000742                          BR      5$              ;CONTINUE TESTING
3328  015526  104306                  7$:     ERROR   306             ;REPORT MORE DATA FROM UNIBUS
3329  015530  000740                          BR      5$              ;CONTINUE WITH TEST
3330  015532  005737  001254          10$:    TST     ERRCNT          ;WERE THERE ANY ERRORS ON THIS TEST?
3331  015536  001401                          BEQ     19$             ;BRANCH IF NO ERRORS ON THIS TEST
3332  015540  104037                          ERROR   37              ;SUMMARY OF UNIBUS ADDRESS FAILURES
3333  015542  012737  015364  001110  19$:    MOV     #20$,$LPERR     ;SET LOOP POINTER TO 20$
3334
3335
3336                          ;;*****************************************************************
3337                          ;*TEST 27       MAIN MEMORY TIMEOUT THROUGH MAP
3338                          ;*
3339                          ;*      THIS TEST GENERATES A TIME OUT THROUGH THE UNIBUS MAP BY TRYING
3340                          ;*      TO REFERENCE ADDRESS 17000000 IN MAIN MEMORY.  IT USES THE LOWEST
3341                          ;*      USEABLE MAP REGISTER, WHICH IN THE DEFAULT CASE IS MAP REGISTER
3342                          ;*      ZERO.
3343                          ;*      NOTE: IF THERE IS MORE THAN 1920K OF MEMORY ON SYSTEM
3344                          ;*      THIS TEST WILL BE SKIPPED.
3345                          ;*
3346                          ;;*****************************************************************
3347  015550                  TST27:
3348  015550  000004                  SCOPE
3349  015552  012737  015676  001324          MOV     #TST30,NXTTST   ;SAVE STARTING ADDRESS OF NEXT TEST
3350                                                                  ;FOR ESCAPE ON PARITY ERRORS
3351  015560  022737  167777  177760          CMP     #167777,SIZELO  ;>1920K?
3352  015566  103002                          BHIS    20$             ;BRANCH IF NOT
3353  015570  000137  015676                  JMP     TST30           ;ELSE SKIP TEST
3354  015574  052737  000040  172516  20$:    BIS     #BIT5,MMR3      ;TURN MAP RELOCATION BACK ON
3355  015602  012737  000020  001264          MOV     #TIMOUT,CPUEXP  ;EXPECTING TIMEOUT IN THIS TEST
3356  015610  013737  001240  172350          MOV     LOWEST,KIPAR4   ;LOAD PAR 4 WITH LOWEST USABLE MAP REG
3357  015616  012777  000074  163422          MOV     #74,@LREGU      ;LOAD UPPER 6 BITS OF LOWEST MAP REG
3358  015624  012777  000000  163412          MOV     #000000,@LREGL  ;LOAD LOWER 16 BITS OF LOWEST MAP REG
3359  015632  012737  015640  001110          MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
3360  015640  005037  001266          1$:     CLR     PCPUER          ;CPU ERROR REG LOCATION
3361  015644  000240                          NOP                     ;THIS IS A SYNC POINT FOR SCOPING
3362  015646  013703  100000                  MOV     @#100000,R3     ;TRY TO READ THRU PAGE 4
3363                                                                  ;THIS REFERENCE WILL GO OUT ON THE
3364                                                                  ;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.
015652  022737  000020  001266          CMP     #TIMOUT,PCPUER  ;THE UNIBUS SHOULD HAVE TIMED OUT
015660  001401                        BEQ     10$             ;BRANCH IF CONDITION WAS CORRECT
015662  104040                        ERROR   40              ;UNIBUS DID NOT TIME OUT
015664  012737  015574  001110  10$:  MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
015672  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.
                                        ;*
                                        ;;*****************************************************************
015676                                  TST30:
                                        SCOPE
015676  000004
015700  012737  016612  001324          MOV     #TST31,NXTTST   ;SAVE STARTING ADDRESS OF NEXT TEST
                                                                ;FOR ESCAPE ON PARITY ERRORS
015706  052737  000040  172516          BIS     #BIT5,MMR3      ;TURN ON MAP RELOCATION
                                        ;;
                                        ;*THE RELOCATION HERE USES A BASE OF 00060000 AND AN
                                        ;*OFFSET OF 00000 TO PRODUCE AN ADDRESS OF 00060000
                                        ;;
015714  012737  015746  001110  100$:  MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
015722  005077  163320                CLR     @LREGU          ;CLEAR UPPER BITS OF MAPPING REG
015726  012777  060000  163310        MOV     #060000,@LREGL  ;LOAD LOWER BITS OF MAPPING REG
015734  013737  001240  172354        MOV     LOWEST,@#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
015742  012700  140000                MOV     #140000,R0      ;SELECT PAR6, OFFSET IS 00000
015746  000240                  1$:   NOP                     ;THIS IS A SYNC POINT FOR SCOPING
015750  011001                        MOV     (R0),R1         ;READ LOCATION 060000 THRU THE UNIBUS
015752  012702  060000                MOV     #060000,R2      ;THE EXPECTED PHYSICAL ADDRESS IS 060000
015756  020102                        CMP     R1,R2           ;SEE IF THE MAP'S FETCH WAS CORRECT
015760  001401                        BEQ     2$              ;BRANCH IF FETCHED DATA MATCHES ADDRESS
015762  104041                        ERROR   41              ;FAULTY RELOCATION BY UNIBUS MAP
                                        ;;
                                        ;*THE RELOCATION HERE USES A BASE OF 00052524 AND AN
                                        ;*OFFSET OF 05252 TO PRODUCE AN ADDRESS OF 00057776
                                        ;;
015764  012737  016016  001110  2$:   MOV     #3$,$LPERR      ;SET LOOP ON ERROR POINTER TO 3$
015772  005077  163250                CLR     @LREGU          ;CLEAR UPPER BITS OF MAPPING REG
015776  012777  052524  163240        MOV     #052524,@LREGL  ;LOAD LOWER BITS OF MAPPING REG
016004  013737  001240  172354        MOV     LOWEST,@#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
016012  012700  145252                MOV     #145252,R0      ;SELECT PAR6, OFFSET IS 05252
016016  000240                  3$:   NOP                     ;THIS IS A SYNC POINT FOR SCOPING
016020  011001                        MOV     (R0),R1         ;READ LOCATION 057776 THRU THE UNIBUS
016022  012702  057776                MOV     #057776,R2      ;THE EXPECTED PHYSICAL ADDRESS IS 057776
016026  020102                        CMP     R1,R2           ;SEE IF THE MAP'S FETCH WAS CORRECT
016030  001401                        BEQ     4$              ;BRANCH IF FETCHED DATA MATCHES ADDRESS
```

```
3421  016032  104041                          ERROR   41              ;FAULTY RFLOCATION BY UNIBUS MAP
3422                                          ;;
3423                                          ;*THE RELOCATION HERE USES A BASE OF 00045252 AND AN
3424                                          ;*OFFSET OF 12524 TO PRODUCE AN ADDRESS OF 00057776
3425                                          ;;
3426  016034  012737  0'6066  001110  4$:     MOV     #5$,$LPERR      ;SET LOOP ON ERROR POINTER TO 5$
3427  016042  005077  163200                 CLR     @LREGU          ;CLEAR UPPER BITS OF MAPPING REG
3428  016046  012777  045252  163170         MOV     #045252,@LREGL  ;LOAD LOWER BITS OF MAPPING REG
3429  016054  013737  001240  172354         MOV     LOWEST,@#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
3430  016062  012700  152524                 MOV     #152524,R0      ;SELECT PAR6, OFFSET IS 12524
3431  016066  000240                  5$:     NOP                     ;THIS IS A SYNC POINT FOR SCOPING
3432  016070  011001                         MOV     (R0),R1         ;READ LOCATION 057776 THRU THE UNIBUS
3433  016072  012702  057776                 MOV     #057776,R2      ;THE EXPECTED PHYSICAL ADDRESS IS 057776
3434  016076  020102                         CMP     R1,R2           ;SEE IF THE MAP'S FETCH WAS CORRECT
3435  016100  001401                         BEQ     6$              ;BRANCH IF FETCHED DATA MATCHES ADDRESS
3436  016102  104041                         ERROR   41              ;FAULTY RELOCATION BY UNIBUS MAP
3437                                          ;;
3438                                          ;*THE RELOCATION HERE USES A BASE OF 00050420 AND AN
3439                                          ;*OFFSET OF 10420 TO PRODUCE AN ADDRESS OF 00061040
3440                                          ;;
3441  016104  012737  016136  001110  6$:     MOV     #7$,$LPERR      ;SET LOOP ON ERROR POINTER TO 7$
3442  016112  005077  163130                 CLR     @LREGU          ;CLEAR UPPER BITS OF MAPPING REG
3443  016116  012777  050420  163120         MOV     #050420,@LREGL  ;LOAD LOWER BITS OF MAPPING REG
3444  016124  013737  001240  172354         MOV     LOWEST,@#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
3445  016132  012700  150420                 MOV     #150420,R0      ;SELECT PAR6, OFFSET IS 10420
3446  016136  000240                  7$:     NOP                     ;THIS IS A SYNC POINT FOR SCOPING
3447  016140  011001                         MOV     (R0),R1         ;READ LOCATION 061040 THRU THE UNIBUS
3448  016142  012702  061040                 MOV     #061040,R2      ;THE EXPECTED PHYSICAL ADDRESS IS 061040
3449  016146  020102                         CMP     R1,R2           ;SEE IF THE MAP'S FETCH WAS CORRECT
3450  016150  001401                         BEQ     8$              ;BRANCH IF FETCHED DATA MATCHES ADDRESS
3451  016152  104041                         ERROR   41              ;FAULTY RELOCATION BY UNIBUS MAP
3452                                          ;;
3453                                          ;*THE RELOCATION HERE USES A BASE OF 00054630 AND AN
3454                                          ;*OFFSET OF 04210 TO PRODUCE AN ADDRESS OF 00061040
3455                                          ;;
3456  016154  012737  016206  001110  8$:     MOV     #9$,$LPERR      ;SET LOOP ON ERROR POINTER TO 9$
3457  016162  005077  163060                 CLR     @LREGU          ;CLEAR UPPER BITS OF MAPPING REG
3458  016166  012777  054630  163050         MOV     #054630,@LREGL  ;LOAD LOWER BITS OF MAPPING REG
3459  016174  013737  001240  172354         MOV     LOWEST,@#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
3460  016202  012700  144210                 MOV     #144210,R0      ;SELECT PAR6, OFFSET IS 04210
3461  016206  000240                  9$:     NOP                     ;THIS IS A SYNC POINT FOR SCOPING
3462  016210  011001                         MOV     (R0),R1         ;READ LOCATION 061040 THRU THE UNIBUS
3463  016212  012702  061040                 MOV     #061040,R2      ;THE EXPECTED PHYSICAL ADDRESS IS 061040
3464  016216  020102                         CMP     R1,R2           ;SEE IF THE MAP'S FETCH WAS CORRECT
3465  016220  001401                         BEQ     10$             ;BRANCH IF FETCHED DATA MATCHES ADDRESS
3466  016222  104041                         ERROR   41              ;FAULTY RELOCATION BY UNIBUS MAP
3467                                          ;;
3468                                          ;*THE RELOCATION HERE USES A BASE OF 00044210 AND AN
3469                                          ;*OFFSET OF 14630 TO PRODUCE AN ADDRESS OF 00061040
3470                                          ;;
3471  016224  012737  016256  001110  10$:    MOV     #11$,$LPERR     ;SET LOOP ON ERROR POINTER TO 11$
3472  016232  005077  163010                 CLR     @LREGU          ;CLEAR UPPER BITS OF MAPPING REG
3473  016236  012777  044210  163000         MOV     #044210,@LREGL  ;LOAD LOWER BITS OF MAPPING REG
3474  016244  013737  001240  172354         MOV     LOWEST,@#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
3475  016252  012700  154630                 MOV     #154630,R0      ;SELECT PAR6, OFFSET IS 14630
3476  016256  000240                  11$:    NOP                     ;THIS IS A SYNC POINT FOR SCOPING
```

```
3477  016260  011001                        MOV     (R0),R1         ;READ LOCATION 061040 THRU THE UNIBUS
3478  016262  012702  061040                MOV     #061040,R2      ;THE EXPECTED PHYSICAL ADDRESS IS 061040
3479  016266  020102                        CMP     R1,R2           ;SEE IF THE MAP'S FETCH WAS CORRECT
3480  016270  001401                        BEQ     12$             ;BRANCH IF FETCHED DATA MATCHES ADDRESS
3481  016272  104041                        ERROR   41              ;FAULTY RELOCATION BY UNIBUS MAP
3482                                ;::
3483                                ;*THE RELOCATION HERE USES A BASE OF 00056734 AND AN
3484                                ;*OFFSET OF 02104 TO PRODUCE AN ADDRESS OF 00061040
3485                                ;::
3486  016274  012737  016326  001110 12$:    MOV     #13$,$LPERR     ;SET LOOP ON ERROR POINTER TO 13$
3487  016302  005077  162740                CLR     @LREGU          ;CLEAR UPPER BITS OF MAPPING REG
3488  016306  012777  056734  162730        MOV     #056734,@LREGL  ;LOAD LOWER BITS OF MAPPING REG
3489  016314  013737  001240  172354        MOV     LOWEST,@#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
3490  016322  012700  142104                MOV     #142104,R0      ;SELECT PAR6, OFFSET IS 02104
3491  016326  000240                13$:    NOP                     ;THIS IS A SYNC POINT FOR SCOPING
3492  016330  011001                        MOV     (R0),R1         ;READ LOCATION 061040 THRU THE UNIBUS
3493  016332  012702  061040                MOV     #061040,R2      ;THE EXPECTED PHYSICAL ADDRESS IS 061040
3494  016336  020102                        CMP     R1,R2           ;SEE IF THE MAP'S FETCH WAS CORRECT
3495  016340  001401                        BEQ     14$             ;BRANCH IF FETCHED DATA MATCHES ADDRESS
3496  016342  104041                        ERROR   41              ;FAULTY RELOCATION BY UNIBUS MAP
3497                                ;::
3498                                ;*THE RELOCATION HERE USES A BASE OF 00042104 AND AN
3499                                ;*OFFSET OF 16734 TO PRODUCE AN ADDRESS OF 00061040
3500                                ;::
3501  016344  012737  016376  001110 14$:    MOV     #15$,$LPERR     ;SET LOOP ON ERROR POINTER TO 15$
3502  016352  005077  162670                CLR     @LREGU          ;CLEAR UPPER BITS OF MAPPING REG
3503  016356  012777  042104  162660        MOV     #042104,@LREGL  ;LOAD LOWER BITS OF MAPPING REG
3504  016364  013737  001240  172354        MOV     LOWEST,@#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
3505  016372  012700  156734                MOV     #156734,R0      ;SELECT PAR6, OFFSET IS 16734
3506  016376  000240                15$:    NOP                     ;THIS IS A SYNC POINT FOR SCOPING
3507  016400  011001                        MOV     (R0),R1         ;READ LOCATION 061040 THRU THE UNIBUS
3508  016402  012702  061040                MOV     #061040,R2      ;THE EXPECTED PHYSICAL ADDRESS IS 061040
3509  016406  020102                        CMP     R1,R2           ;SEE IF THE MAP'S FETCH WAS CORRECT
351C  016410  001401                        BEQ     16$             ;BRANCH IF FETCHED DATA MATCHES ADDRESS
3511  016412  104041                        ERROR   41              ;FAULTY RELOCATION BY UNIBUS MAP
3512                                ;::
3513                                ;*THE RELOCATION HERE USES A BASE OF 00057776 AND AN
3514                                ;*OFFSET OF 01042 TO PRODUCE AN ADDRESS OF 00061040
3515                                ;::
3516  016414  012737  016446  001110 16$:    MOV     #17$,$LPERR     ;SET LOOP ON ERROR POINTER TO 17$
3517  016422  005077  162620                CLR     @LREGU          ;CLEAR UPPER BITS OF MAPPING REG
3518  016426  012777  057776  162610        MOV     #057776,@LREGL  ;LOAD LOWER BITS OF MAPPING REG
3519  016434  013737  001240  172354        MOV     LOWEST,@#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
3520  016442  012700  141042                MOV     #141042,R0      ;SELECT PAR6, OFFSET IS 01042
3521  016446  000240                17$:    NOP                     ;THIS IS A SYNC POINT FOR SCOPING
3522  016450  011001                        MOV     (R0),R1         ;READ LOCATION 061040 THRU THE UNIBUS
3523  016452  012702  061040                MOV     #061040,R2      ;THE EXPECTED PHYSICAL ADDRESS IS 061040
3524  016456  020102                        CMP     R1,R2           ;SEE IF THE MAP'S FETCH WAS CORRECT
3525  016460  001401                        BEQ     18$             ;BRANCH IF FETCHED DATA MATCHES ADDRESS
3526  016462  104041                        ERROR   41              ;FAULTY RELOCATION BY UNIBUS MAP
3527                                ;::
3528                                ;*THE RELOCATION HERE USES A BASE OF 00041042 AND AN
3529                                ;*OFFSET OF 17776 TO PRODUCE AN ADDRESS OF 00061040
3530                                ;::
3531  016464  012737  016516  001110 18$:    MOV     #19$,$LPERR     ;SET LOOP ON ERROR POINTER TO 19$
3532  016472  005077  162550                CLR     @LREGU          ;CLEAR UPPER BITS OF MAPPING REG
```

```
3533  016476  012777  041042  162540          MOV     #041042,@LREGL  ;LOAD LOWER BITS OF MAPPING REG
3534  016504  013737  001240  172354          MOV     LOWEST,@#KIPAR6 ;LOAD PAR6 WITH ADDR O. LOWEST MAP REG
3535  016512  012700  157776                  MOV     #157776,R0      ;SELECT PAR6, OFFSET IS 17776
3536  016516  000240                  19$:    NOP                     ;THIS IS A SYNC POINT FOR SCOPING
3537  016520  011001                          MOV     (R0),R1         ;READ LOCATION 061040 THRU THE UNIBUS
3538  016522  012702  061040                  MOV     #061040,R2      ;THE EXPECTED PHYSICAL ADDRESS IS 061040
3539  016526  020102                          CMP     R1,R2           ;SEE IF THE MAP'S FETCH WAS CORRECT
3540  016530  001401                          BEQ     20$             ;BRANCH IF FETCHED DATA MATCHES ADDRESS
3541  016532  104041                          ERROR   41              ;FAULTY RELOCATION BY UNIBUS MAP
3542                                  ;;
3543                                  ;*THE RELOCATION HERE USES A BASE OF 00057776 AND AN
3544                                  ;*OFFSET OF 00002 TO PRODUCE AN ADDRESS OF 00060000
3545                                  ;;
3546  016534  012737  016566  001110  20$:    MOV     #21$,$LPERR     ;SET LOOP ON ERROR POINTER TO 21$
3547  016542  005077  162500                  CLR     @LREGU          ;CLEAR UPPER BITS OF MAPPING REG
3548  016546  012777  057776  162470          MOV     #057776,@LREGL  ;LOAD LOWER BITS OF MAPPING REG
3549  016554  013737  001240  172354          MOV     LOWEST,@#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
3550  016562  012700  140002                  MOV     #140002,R0      ;SELECT PAR6, OFFSET IS 00002
3551  016566  000240                  21$:    NOP                     ;THIS IS A SYNC POINT FOR SCOPING
3552  016570  011001                          MOV     (R0),R1         ;READ LOCATION 060000 THRU THE UNIBUS
3553  016572  012702  060000                  MOV     #060000,R2      ;THE EXPECTED PHYSICAL ADDRESS IS 060000
3554  016576  020102                          CMP     R1,R2           ;SEE IF THE MAP'S FETCH WAS CORRECT
3555  016600  001401                          BEQ     22$             ;BRANCH IF FETCHED DATA MATCHES ADDRESS
3556  016602  104041                          ERROR   41              ;FAULTY RELOCATION BY UNIBUS MAP
3557  016604  012737  015714  001110  22$:    MOV     #100$,$LPERR    ;SET LOOP POINTER TO START OF TEST
3558
3559                                  ;;**********************************************************************
3560                                  ;*TEST 31        TEST CARRY PROPAGATION OF MAP'S RELOCATION ADDER
3561                                  ;*
3562                                  ;*      EVERY ADDRESS OF THE FORM XXXX0000 IS GENERATED HERE STARTING
3563                                  ;*      WITH 00030000 UP TO 17000000.  THAT IS, THE FIRST OF EVERY 2K
3564                                  ;*      WORDS IS ADDRESSED. TO INSURE THAT THE ADDER IN THE MAP IS
3565                                  ;*      WORKING PROPERLY AND, THE SYSTEM SIZE JUMPERS ARE ACTUALLY SET
3566                                  ;*      FOR THE TOP OF MAIN MEMORY.
3567                                  ;*
3568                                  ;**********************************************************************
3569
3570  016612                  TST31:
3571  016612  000004                  SCOPE
3572  016614  012737  017116  001324          MOV     #TST32,NXTTST   ;SAVE STARTING ADDRESS OF NEXT TEST
3573                                                                  ;FOR ESCAPE ON PARITY ERRORS
3574  016622  012737  000012  001204          MOV     #12,$TIMES      ;;DO 12 ITERATIONS
3575                                          .EQUIV  BIT5,NEXMEM     ;BIT05 IS NON-EXISTANT MEMORY BIT IN
3576                                                                  ;THE CPU ERROR REGISTER
3577  016630  012737  177777  016764  20$:    MOV     #-1,4$          ;INITIALIZE FLAG AS NEGATIVE ONE
3578  016636  005077  162404                  CLR     @LREGU          ;CLEAR UPPER 6 BITS OF MAP REG
3579  016642  012777  020000  162374          MOV     #20000,@LREGL   ;LOAD 4K BASE INTO MAP REGISTER
3580  016650  012701  100100                  MOV     #100100,R1      ;LOAD BITS TO SELECT PAR 4. OFFSET 100
3581  016654  012700  150000                  MOV     #150000,R0      ;LOAD BITS TO SELECT PAR 6. OFFSET 2K
3582  016660  012737  000277  172350          MOV     #277,KIPAR4     ;START WITH PHYSICAL 6K
3583  016666  013737  001240  172354          MOV     LOWEST,KIPAR6   ;LOAD PAR 6 WITH MAP REG'S ADDR
3584  016674  012737  016734  001110          MOV     #10$,$LPERR     ;SET LOOP ON ERROR POINTER TO 10$
3585  016702  012737  000020  001264  3$:     MOV     #20,CPUEXP      ;EXPECTING A UNIBUS TIME OUT DURING TEST
3586  016710  005037  001266                  CLR     PCPUER          ;CLEAR TIME OUT FLAG
3587  016714  013710  001326                  MOV     DATA,(R0)       ;THIS LOAD WILL TIME OUT WHEN YOU
3588                                                                  ;HAVE REACHED THE TOP OF MEMORY
```

```
3589                                                    ;IT SELECTS PAR 6 WHICH WILL PUT ADDR
3590                                                    ;<XXX XX1>0000 ON THE UNIBUS.
3591                                                    ;THE X'S WILL SELECT THE LOWEST USEABLE
3592                                                    ;MAPPING REGISTER.  THE DEFAULT CASE IS
3593                                                    ;010000,  SELECTING MAPPING REGISTER 0.
3594  016720  005737  001266                  TST     PCPUER          ;SEE IF THERE WAS MAIN MEMORY
3595  016724  001016                          BNE     1$              ;BRANCH IF NO MAIN MEMORY FROM UNIBUS
3596  016726  012737  000040  001264          MOV     #NEXMEM,CPUEXP  ;POSSIBLE CACHE NON-EXISTANT MEMORY
3597  016734  011103                  10$:    MOV     (R1),R3         ;READ TEST LOCATION VIA FASTBUS
3598  016736  022737  000040  001266          CMP     #NEXMEM,PCPUER  ;WAS THIS CACHE NON-EXISTANT MEMORY
3599  016744  001414                          BEQ     2$              ;BRANCH IF NON-EXSITANT MEMORY
3600  016746  000240                          NOP                     ;THIS IS A SYNC POINT FOR SCOPING
3601  016750  011002                          MOV     (R0),R2         ;READ TEST LOCATION VIA UNIBUS MAP
3602  016752  020203                          CMP     R2,R3           ;COMPARE TEST DATA R2=MAP DATA
3603                                                                  ;R3=FASTBUS DATA
3604  016754  001410                          BEQ     2$              ;BRANCH IF IT WAS THE SAME
3605  016756  104042                          ERROR   42              ;CARRY PROPAGATION FAILURE IN MAP
3606  016760  000406                          BR      2$              ;BRANCH TO UPDATE ROUTINE
3607  016762  005227                  1$:     INC     (PC)+           ;INCREMENT ONE TIME ENTRANCE FLAG
3608  016764  177777                  4$:     .WORD   -1              ;USE NEGATIVE ONE FOR FLAG
3609  016766  001003                          BNE     2$              ;BRANCH IF YOU'VE BEEN HERE BEFORE
3610  016770  013737  172350  001320          MOV     KIPAR4,RSIZE    ;SAVE UPPER LIMIT OF MEMORY
3611  016776  062737  000100  001326  2$:     ADD     #100,DATA       ;CHANGE PATTERN FOR NEXT LOAD
3612  017004  062737  000100  172350          ADD     #100,KIPAR4     ;ADD 2K TO PAR4
3613  017012  062777  010000  162224          ADD     #10000,@LREGL   ;ADD 2K TO MAP REGISTER
3614  017020  001330                          BNE     3$              ;BRANCH IF MAP REGISTER NOT ZERO
3615  017022  005277  162220                  INC     @LREGU          ;ADD ONE TO UPPER 6 BITS OF MAP REG
3616  017026  022777  000073  162212          CMP     #73,@LREGU      ;SEE IF TOP 128K BLOCK HAS BEEN PASSED
3617  017034  103322                          BHIS    3$              ;BRANCH IF NOT PAST IT
3618  017036  005237  001326                  INC     DATA            ;CHANGE DATA PATTERN FOR NEXT PASS
3619  017042  042737  177700  001326          BIC     #177700,DATA    ;CLEAR UPPER 10 BITS OF DATA PATTERN
3620  017050  052737  000300  001326          BIS     #300,DATA       ;START WITH 3XX IN DATA PATTERN
3621  017056  012737  016630  001110          MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
3622  017064  023737  177760  001320          CMP     @#SIZELO,RSIZE  ;SEE IF SIZE JUMPERS AGREE WITH MEMORY TOP
3623  017072  001407                          BEQ     19$             ;BRANCH IF TOP OF MEMORY AGREES WITH
3624                                                                  ;THE SIZE JUMPERS
3625  017074  105737  001331                  TSTB    KB11EM          ;RUNNING ON A KB11-EM?
3626  017100  001004                          BNE     19$             ;SKIP ERROR IF YES
3627  017102  105737  001332                  TSTB    KB11CM          ;RUNNING ON MODIFIED MACHINE?
3628  017106  001001                          BNE     19$             ;BRANCH IF YES
3629  017110  104043                          ERROR   43              ;TOP OF MEMORY DOESN'T MATCH SIZE JUMPER
3630  017112  005037  001264          19$:    CLR     CPUEXP          ;NO CPU TRAPS EXPECTED FOR AWHILE
3631
3632
3633                                  ;*****************************************************************
3634                                  ;*TEST 32       PARITY REPORTING THRU THE MAP, MAIN MEMORY EVEN WORD
3635                                  ;*
3636                                  ;*      THIS TEST FORCES A PARITY ERROR IN MAIN MEMORY AT PHYSICAL
3637                                  ;*      ADDRESS 00040000 BY SETTING ONE BIT IN EACH BYTE OF THE
3638                                  ;*      WORD.  THE TEST THEN FORCES THE EVEN WORD PARITY BITS TO
3639                                  ;*      ONES AND READS ADDRESS 00040000 THRU THE UNIBUS MAP.  THEN
3640                                  ;*      IT CHECKS TO SEE THAT THE PARITY ABORT OCCURRED AND THAT THE
3641                                  ;*      CONDITION WAS CORRECT BEFORE GOING TO THE NEXT TEST.
3642                                  ;*
3643                                  ;*      ERRORS ARE REPORTED INLINE IN THE TEST CODE, AND THE PARITY
3644
```

```
3645                                    ;*      VECTOR WILL TRAP TO LABEL 10$ AT THE END OF THIS TEST.
3646                                    ;*
3647                                    ;**********************************************************************
3648  017116                            TST32:
3649  017116  000004                           SCOPE
3650  017120  012737  017360  001324           MOV     #1ST33,NXTTST   ;SAVE STARTING ADDRESS OF NEXT TEST
3651                                                                    ;FOR ESCAPE ON PARITY ERRORS
3652  017126  104416                           TBITO                   ;TURN OFF T BIT TRAPPING IF ON
3653  017130  012737  023404  001200  20$:     MOV     #23404,$TMP4    ;EXPECTED ERROR CONDITION
3654  017136  005037  001274                   CLR     PPARER          ;CLEAR MEM ERROR REG STORAGE
3655  017142  012737  177777  177744           MOV     #-1,@#MEMERR    ;CLEAR MEMORY ERROR REGISTER
3656  017150  013737  177746  001202           MOV     @#CONTRL,$TMP5  ;SAVE CONTROL REG TO RESTORE CACHE
3657  017156  012737  000014  177746           MOV     #14,@#CONTRL    ;FORCE MISSES IN CACHE
3658  017164  013737  001240  172354           MOV     LOWEST,KIPAR6   ;PUT UNIBUS ADDR OF MAP REG IN PAR6
3659  017172  012777  040000  162044           MOV     #40000,@LREGL   ;LOWEST MAP REGISTER POINTS TO 8K
3660  017200  005077  162042                   CLR     @LREGU          ;CLEAR UPPER BITS OF MAP REG
3661  017204  012701  177750                   MOV     #MAINT,R1       ;PUT ADDR OF MAINT REG IN R1
3662  017210  012737  017234  001110           MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
3663  017216  012700  140000                   MOV     #140000,R0      ;SELECT KIPAR6, OFFSET 0
3664  017222  012710  020001                   MOV     #20001,(R0)     ;LOAD ONE BIT IN EACH BYTE OF 40000
3665  017226  012737  017320  000114           MOV     #10$,CACHVEC    ;SET PARITY VECTOR TO 10$
3666  017234  012737  030000  177750  1$:      MOV     #30000,@#MAINT  ;FORCE EVEN WORD PARITY BITS TO ONE
3667  017242  000240                           NOP                     ;THIS IS A SYNC POINT FOR SCOPING
3668  017244  011003                           MOV     (R0),R3         ;'DATA FETCH' SHOULD CAUSE PARITY ABORT
3669  017246  011102                           MOV     (R1),R2         ;SAVE MAINT REG IN CASE NO TRAP
3670  017250  005011                           CLR     (R1)            ;CLEAR MAINT REG IN CASE NO TRAP
3671  017252  050000                           BIS     R0,R0           ;DUMMY INST WITH P.B.'S ON
3672  017254  023737  001200  001274           CMP     $TMP4,PPARER    ;SEE IF ERROR REG HOLDS RIGHT DATA
3673  017262  001401                           BEQ     2$              ;BRANCH IF CONDITION WAS CORRECT
3674  017264  104044                           ERROR   44              ;PARITY REPORTING BAD
3675  017266  012737  177777  177744  2$:      MOV     #-1,MEMERR      ;CLEAR MEMORY ERROR REGISTER
3676  017274  012737  005764  000114           MOV     #MEMER,CACHVEC  ;RESTORE PARITY SERVICE ROUTINE
3677  017302  013737  001202  177746           MOV     $TMP5,CONTRL    ;RESTORE CACHE TO FORMER CONDITION
3678  017310  012737  017130  001110           MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
3679  017316  000420                           BR      TST33           ;;TEST OVER GO TO NEXT TEST
3680
3681
3682  017320  013737  177740  001270  10$:     MOV     @#LOADRS,PLOADR ;SAVE LOWER CACHE ADDR REG
3683  017326  013737  177742  001272           MOV     @#HIADRS,PHIADR ;SAVE HIGH BITS OF FAILING ADDR
3684  017334  013737  177744  001274           MOV     @#MEMERR,PPARER ;SAVE MEMORY ERROR REGISTER
3685  017342  013737  177746  001276           MOV     @#CONTRL,PCONTR ;SAVE CONTROL REGISTER FOR TYPE OUT
3686  017350  012737  030000  001300           MOV     #30000,PMAINT   ;SAVE DATA IN MAINTENENCE REGISTER
3687  017356  000002                           RTI                     ;RETURN TO TEST AND CHECK PARITY
3688
3689
3690                                    ;**********************************************************************
3691                                    ;*TEST 33       PARITY REPORTING THRU THE MAP, MAIN MEMORY ODD WORD
3692                                    ;*
3693                                    ;*      THIS TEST FORCES A PARITY ERROR IN MAIN MEMORY AT PHYSICAL
3694                                    ;*      ADDRESS 00040002 BY SETTING ONE BIT IN EACH BYTE OF THE
3695                                    ;*      WORD.  THE TEST THEN FORCES THE ODD WORD PARITY BITS TO
3696                                    ;*      ONES AND READS ADDRESS 00040002 THRU THE UNIBUS MAP.  THEN
3697                                    ;*      IT CHECKS TO SEE THAT THE PARITY ABORT OCCURRED AND THAT THE
3698                                    ;*      CONDITION WAS CORRECT BEFORE GOING TO THE NEXT TEST.
3699                                    ;*
3700                                    ;*      ERRORS ARE REPORTED INLINE IN THE TEST CODE, AND THE PARITY
```

```
                                        ;*      VECTOR WILL TRAP TO LABEL 10$ AT THE END OF THIS TEST.
                                        ;*
                                        ;*********************************************************************
017360                                  TST33:
                                                SCOPE
017360  000004
017362  012737  017632  001324                  MOV     #TST34,NXTTST   ;SAVE STARTING ADDRESS OF NEXT TEST
                                                                        ;FOR ESCAPE ON PARITY ERRORS
017370  012737  023410  001200  20$:     MOV     #23410,$TMP4    ;LOAD EXPECTED ERROR CONDITION
017376  005037  001274                  CLR     PPARER          ;CLEAR MEM ERROR REG STORAGE
017402  012737  177777  177744          MOV     #-1,@#MEMERR    ;CLEAR MEMORY ERROR REGISTER
017410  013737  177746  001202          MOV     @#CONTRL,$TMP5  ;SAVE CONTROL REG TO RESTORE CACHE
017416  012737  000014  177746          MOV     #14,@#CONTRL    ;FORCE MISSES IN CACHE
017424  013737  001240  172354          MOV     LOWEST,KIPAR6   ;PUT UNIBUS ADDR OF MAP REG IN PAR6
017432  012777  040000  161604          MOV     #40000,@LREGL   ;LOWEST MAP REGISTER POINTS TO 8K
017440  005077  161602                  CLR     @LREGU          ;CLEAR UPPER BITS OF MAP REG
017444  012701  177750                  MOV     #MAINT,R1       ;PUT ADDR OF MAINT REG IN R1
017450  012737  017506  001110          MOV     #1$,$LPERR      ;SET LOOP ON EROR POINTER TO 1$
017456  005037  140000                  CLR     140000          ;CLEAR LOCATION 40000
017462  012700  140002                  MOV     #140002,R0      ;LOAD ADDRESS OF 40002 INTO R0
017466  012737  017506  001110          MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
017474  012710  020001                  MOV     #20001,(R0)     ;LOAD ONE BIT IN EACH BYTE OF 40002
017500  012737  017572  000114          MOV     #10$,CACHVEC    ;SET PARITY VECTOR TO 10$
017506  012737  140000  177750  1$:     MOV     #140000,@#MAINT ;FORCE ODD WORD PARITY BITS TO ONE
017514  000240                          NOP                     ;THIS IS A SYNC POINT FOR SCOPING
017516  011003                          MOV     (R0),R3         ;'DATA FETCH' SHOULD CAUSE PARITY ABORT
017520  011102                          MOV     (R1),R2         ;SAVE MAINT REG IN CASE NO TRAP
017522  005011                          CLR     (R1)            ;CLEAR MAINT REG IN CASE NO TRAP
017524  050000                          BIS     R0,R0           ;DUMMY INST WITH P.B.'S ON
017526  023737  001200  001274          CMP     $TMP4,PPARER    ;SEE IF MEM ERROR REG HOLDS RIGHT DATA
017534  001401                          BEQ     2$              ;BRANCH IF ERROR WAS EXPECTED ONE
017536  104044                          ERROR   44              ;PARITY REPORTING BAD
017540  012737  177777  177744  2$:     MOV     #-1,MEMERR      ;CLEAR MEMORY ERROR REGISTER
017546  012737  005764  000114          MOV     #MEMER,CACHVEC  ;RESTORE PARITY SERVICE ROUTINE
017554  013737  001202  177746          MOV     $TMP5,CONTRL    ;RESTORE CACHE TO FORMER CONDITION
017562  012737  017370  001110          MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
017570  000420                          BR      TST34           ;;TEST IS OVER, GO TO NEXT TEST
                                        ;
                                        ;
017572  013737  177740  001270  10$:     MOV     @#LOADRS,PLOADR ;SAVE LOWER CACHE ADDR REG
017600  013737  177742  001272          MOV     @#HIADRS,PHIADR ;SAVE HIGH BITS OF FAILING ADDR
017606  013737  177744  001274          MOV     @#MEMERR,PPARER ;SAVE MEMORY ERROR REGISTER
017614  013737  177746  001276          MOV     @#CONTRL,PCONTR ;SAVE CONTROL REGISTER FOR TYPE OUT
017622  012737  140000  001300          MOV     #140000,PMAINT  ;SAVE DATA IN MAINTENENCE REGISTER
017630  000002                          RTI                     ;RETURN TO TEST AND CHECK PARITY
                                        ;
                                        ;
                                        ;*********************************************************************
                                        ;*TEST 34       PARITY REPORTING THRU THE MAP, CACHE DATA GROUP 0
                                        ;*
                                        ;*      THIS TEST FORCES A PARITY ERROR IN CACHE GROUP 0 AT PHYSICAL
                                        ;*      ADDRESS 00040010 BY SETTING BIT 15 IN THE WORD.  THE TEST
                                        ;*      THEN FORCES GROUP 0 LOW BYTE PARITY BIT TO ZERO AND EVEN
                                        ;*      WORD HIGH BYTE PARITY BIT TO ONE BEFORE READING ADDRESS 00040010
                                        ;*      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.
                        ;*
                        ;**************************************************************************
017632                  TST34:
                                SCOPE
017632  000004
017634  012737  020120  001324         MOV     #TST35,NXTTST   ;SAVE STARTING ADDRESS OF NEXT TEST
                                                                ;FOR ESCAPE ON PARITY ERRORS
017642  012737  177777  020006  20$:    MOV     #-1,21$         ;RESTORE NEG ONE FOR NEXT ITERATION
017650  032737  000004  177746         BIT     #BIT2,@#CONTRL  ;SEE IF GROUP 0 IS DISABLED
017656  001120                         BNE     TST35           ;;BRANCH IF GROUP 0 IS DISABLED
017660  013737  177746  001202         MOV     CONTRL,$TMP5    ;SAVE CONDITION OF CACHE
017666  012737  023504  001200         MOV     #23504,$TMP4    ;EXPECTED PARITY CONDITION GROUP 0
017674  005037  001274                 CLR     PPARER          ;CLEAR MEM ERROR REG STORAGE
017700  012737  177777  177744         MOV     #-1,@#MEMERR    ;CLEAR MEMORY ERROR REGISTER
017706  013737  001240  172354         MOV     LOWEST,KIPAR6   ;PUT UNIBUS ADDR OF MAP REG IN PAR6
017714  005077  161326                 CLR     @LREGU          ;CLEAR UPPER BITS ON MAP REG
017720  012777  040010  161316         MOV     #40010,@LREGL   ;LOAD LOWER 16 BITS OF MAP REG
017726  012705  177750                 MOV     #MAINT,R5       ;PUT ADDR OF MAINT REG IN R5
017732  012704  140000                 MOV     #140000,R4      ;SELECT PAR 6 BASE 0
017736  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 TO BE OFF
017742  012737  000030  177746         MOV     #30,@#CONTRL    ;FORCE SELECTION OF GROUP 0 ON READ
017750  011401                         MOV     (R4),R1         ;PUT (40000) & (40002) IN GROUP 0
017752  005001                         CLR     R1              ;ZERO R1 FOR THE MOMENT
017754  012737  020060  000114         MOV     #10$,CACHVEC    ;SET PARITY VECTOR TO 10$
017762  010137  177750          1$:    MOV     R1,@#MAINT      ;FORCE GPO PARITY BITS TO 0 ON 2ND PASS
017766  000241                         CLC                     ;THIS IS A SYNC POINT FOR SCOPING
                                                                ;THE LOW BYTE HAS ITS P.B. OFF
017770  011401                         MOV     (R4),R1         ;DATA FETCH SHOULD CAUSE PARITY ABORT
017772  011503                         MOV     (R5),R3         ;SAVE MAINT REG IN CASE NO TRAP
017774  1C                             CLRB    (R5)            ;CLEAR MAINT REG IN CASE NO TRAP
017776  006.                           SXT     R1              ;DUMMY INST WITH P.B.'S OFF
020000  012701  0..020                MOV     #020020,R1      ;FORCE GROUP 0 LOW BYTE PARITY BIT
                                                                ;TO 0 AND MAIN MEMORY EVEN WORD HIGH
                                                                ;BYTE PARITY BIT TO 1
020004  005227                         INC     (PC)+           ;INCREMENT NEXT WORD TO ZERO ON FIRST PASS
020006  177777          21$:    .WORD   -1                      ;
020010  001764                         BEQ     1$              ;BRANCH ON FIRST PASS ONLY
020012  023737  001200  001274         CMP     $TMP4,PPARER    ;SEE IF ERROR CONDITION MATCHES EXPECTED
020020  001402                         BEQ     2$              ;BRANCH IF CORRECT CONDITION
020022  010302                         MOV     R3,R2           ;R2 HOLDS MAINT REG FOR ERROR TYPE OUT
020024  104044                         ERROR   44              ;PARITY REPORTING BAD
020026  012737  177777  177744  2$:    MOV     #-1,MEMERR      ;CLEAR MEMORY ERROR REGISTER
020034  012737  005764  000114         MOV     #MEMER,CACHVEC  ;RESTORE PARITY SERVICE ROUTINE
020042  013737  001202  177746         MOV     $TMP5,CONTRL    ;RESTORE CACHE TO FORMER CONDITION
020050  012737  017642  001110         MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
020056  000420                         BR      TST35           ;;TEST OVER GO TO NEXT TEST

020060  013737  177740  001270  10$:    MOV     @#LOADRS,PLOADR ;SAVE LOWER CACHE ADDR REG
020066  013737  177742  001272         MOV     @#HIADRS,PHIADR ;SAVE HIGH BITS OF FAILING ADDR
020074  013737  177744  001274         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
                        MOV     #44,@#CONTRL    ;FORCE SELECTION OF GROUP 0 ON READ
                        MOV     (R4),R1         ;PUT (40000) & (40002) IN GROUP 1
                        CLR     R1              ;CLEAR R1 FOR MOMENT
                        MOV     #10$,CACHVEC    ;SET PARITY VECTOR TO 10$
1$:                     MOV     R1,@#MAINT      ;FORCE GP1 PARITY BITS TO 0 ON 2ND PASS
                        CLC                     ;THIS IS A SYNC POINT FOR SCOPING
                                                ;THE LOW BYTE HAS ITS P.B. OFF
                        MOV     (R4),R1         ;DATA FETCH SHOULD CAUSE PARITY ABORT
                        MOV     (R5),R3         ;SAVE MAINT REG IN CASE NO TRAP
                        CLRB    (R5)            ;CLEAR MAINT REG IN CASE NO TRAP
                        SXT     R1              ;DUMMY INST WITH P.B.'S OFF
                        MOV     #020100,R1      ;FORCE GROUP 1 LOW BYTE PARITY BIT
                                                ;TO 0 AND MAIN MEMORY EVEN WORD HIGH
                                                ;BYTE PARITY BIT TO 1
                        INC     (PC)+           ;INCREMENT NEXT WORD TO 0
21$:                    .WORD   -1              ;
```

```
3869  020276  001764                          BEQ     1$                  ;BRANCH ON FIRST PASS ONLY
3870  020300  023737  001200  001274          CMP     $TMP4,PPARER        ;SEE IF ERROR CONDITION MATCHES EXPECTED
3871  020306  001402                          BEQ     2$                  ;BRANCH IF ERROR WAS EXPECTED ONE
3872  020310  010302                          MOV     R3,R2               ;R2 HOLDS MAINT REG FOR ERROR TYPE OUT
3873  020312  104044                          ERROR   44                  ;PARITY REPORTING BAD
3874  020314  012737  177777  177744  2$:     MOV     #-1,MEMERR          ;CLEAR MEMORY ERROR REGISTER
3875  020322  012737  005764  000114          MOV     #MEMER,CACHVEC      ;RESTORE PARITY SERVICE ROUTINE
3876  020330  013737  001202  177746          MOV     $TMP5,CONTRL        ;RESTORE CACHE TO FORMER CONDITION
3877  020336  012737  020130  001110          MOV     #20$,$LPERR         ;SET LOOP POINTER TO START OF TEST
3878  020344  000420                          BR      UBMAP               ;;TEST OVER GO TO NEXT TEST
3879
3880
3881  020346  013737  177740  001270  10$:    MOV     @#LOADRS,PLOADR     ;SAVE LOWER CACHE ADDR REG
3882  020354  013737  177742  001272          MOV     @#HIADRS,PHIADR     ;SAVE HIGH BITS OF FAILING ADDR
3883  020362  013737  177744  001274          MOV     @#MEMERR,PPARER     ;SAVE MEMORY ERROR REGISTER
3884  020370  013737  177746  001276          MOV     @#CONTRL,PCONTR     ;SAVE CONTROL REGISTER FOR TYPE OUT
3885  020376  012737  030300  001300          MOV     #30300,PMAINT       ;SAVE DATA IN MAINTENENCE REGISTER
3886  020404  000002                          RTI                         ;RETURN TO TEST AND CHECK PARITY
3887
3888
3889                                  .SBTTL  ************************************************************
3890                                  .SBTTL  THE FOLLOWING TESTS ARE RUN THROUGH THE UNIBUS MAP
3891                                  .SBTTL  ************************************************************
3892
3893
3894
3895  020406  000004                  UBMAP:  SCOPE                       ;LOOP ON PREVIOUS TEST
3896  020410  104420                          TBITR                       ;RESTORE T-BIT IF IT WAS ON
3897                                  ;*
3898                                  ;*      THIS CODE SETS UP THE TWO MAP REGISTERS ABOVE THE LOWEST
3899                                  ;*      USEABLE ONE TO POINT TO PHYSICAL MEMORY FROM 0 - 8K.  IN THE
3900                                  ;*      DEFAULT CASE, IN MANUFACTURING AND IF THERE IS NO UNIBUS MEMORY,
3901                                  ;*      THIS WILL BE MAP REGISTERS 1 AND 2.  MAP REGISTER 1 WILL POINT
3902                                  ;*      TO PHYSICAL 4K - 8K SO "ACT-11" WILL WORK PROPERLY AND MAP
3903                                  ;*      REGISTER 2 WILL POINT TO PHYSICAL 0 - 4K.  THIS MEANS THAT
3904                                  ;*      KIPAR0 SHOULD GET 170400 SO IT PUTS ADDRESSES 040000 TO 057776
3905                                  ;*      ON THE UNIBUS AND KIPAR1 SHOULD GET 170200 SO IT PUTS ADDRESSES
3906                                  ;*      020000 TO 037776 ON THE UNIBUS.
3907                                  ;*
3908  020412  013701  001246                  MOV     LREGU,R1            ;PUT POINTER TO LOWEST MAP REG IN R1
3909  020416  005061  000004                  CLR     4(R1)               ;CLEAR UPPER 6 BITS OF (LOWEST + 1)
3910  020422  005061  000010                  CLR     10(R1)              ;CLEAR UPPER 6 BITS OF (LOWEST + 2)
3911  020426  012761  020000  000002          MOV     #20000,2(R1)        ;LOAD LOWER 16 BITS OF (LOWEST + 1)
3912                                                                      ;SO THAT IT POINTS TO 4K - 8K
3913  020434  005061  000006                  CLR     6(R1)               ;CLEAR LOWER 16 BITS OF (LOWEST + 2)
3914  020440  013701  001240                  MOV     LOWEST,R1           ;PREPARE TO LOAD PAR1
3915  020444  062701  000200                  ADD     #200,R1             ;POINTER TO (LOWEST + 1)
3916  020450  010137  172342                  MOV     R1,KIPAR1           ;LOAD PAR1 TO POINT TO LOWEST + 1
3917  020454  062701  000200                  ADD     #200,R1             ;ADJUST R1 FOR KIPAR0
3918  020460  010137  172340                  MOV     R1,KIPAR0           ;LOAD PAR0 TO POINT TO (LOWEST + 2)
3919
3920
3921                                  ;;*************************************************************
3922                                  ;*TEST 36       MAIN MEMORY TIMEOUT THRU MAP, CODE RUN OVER UNIBUS
3923                                  ;*
3924                                  ;*      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.
                                        ;*
                                        ;*      THIS TEST IS BEING RUN WITH ALL MEMORY REFERENCES GOING THRU
                                        ;*      THE UNIBUS MAP.
                                        ;*      NOTE: IF THERE IS MORE THAN 1920K OF MEMORY ON SYSTEM
                                        ;*      THIS TEST WILL BE SKIPPED.
                                        ;*
                                        ;**************************************************************
020464                                  TST36:
                                        SCOPE
020464  000004                          
020466  012737  020540  001106         MOV     #20$,$LPADR     ;SET LOOP ON TEST POINTER TO 20$
020474  012737  020540  001110         MOV     #20$,$LPERR     ;SET LOOP ON ERROR POINTER TO 20$
020502  012737  000036  001102         MOV     #36,$TSTNM      ;SETUP TEST NUMBER AND CLR ERROR FLAG
020510  013737  001102  177570         MOV     $TSTNM,DISPLAY  ;DISPLAY TEST NUMBER FOR ALL TO SEE
020516  012737  020634  001324         MOV     #TST37,NXTTST   ;SET UP ESCAPE VECTOR IN CASE OF PARITY ERRORS
020524  022737  167777  177760         CMP     #167777,SIZELO  ;>1920K?
020532  103002                         BHIS    20$             ;BRANCH IF NOT
020534  000137  020634                 JMP     TST37           ;ELSE SKIP TEST
020540  012737  000020  001264  20$:    MOV     #20,CPUEXP      ;EXPECTING CPU TIMEOUT IN THIS TEST
020546  013737  001240  172350         MOV     LOWEST,KIPAR4   ;LOAD PAR 4 WITH LOWEST USABLE MAP REG
020554  012777  000074  160464         MOV     #74,@LREGU      ;LOAD UPPER 6 BITS OF LOWEST MAP REG
020562  012777  000000  160454         MOV     #000000,@LREGL  ;LOAD LOWER 16 BITS OF LOWEST MAP REG
020570  012737  020576  001110         MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
020576  005037  001266          1$:    CLR     PCPUER          ;CPU ERROR REG LOCATION
020602  000240                         NOP                     ;THIS IS A SYNC POINT FOR SCOPING
020604  013703  100000                 MOV     @#^O100000,R3   ;TRY TO READ THRU PAGE 4
                                                                ;THIS REFERENCE WILL GO OUT ON THE
                                                                ;UNIBUS TO SELECT THE LOWEST USEABLE
                                                                ;MAP REGISTER (DEFAULT MAP REG. 0).
                                                                ;PHYSICAL ADDRESS 17700000 IS THEN
                                                                ;GENERATED, WHICH SHOULD TIME OUT SINCE
                                                                ;IT IS THE FIRST NON-EXISTANT LOCATION.
020610  022737  000020  001266         CMP     #20,PCPUER      ;THE UNIBUS SHOULD HAVE TIMED OUT
020616  001401                         BEQ     10$             ;BRANCH IF UNIBUS TIMED OUT
020620  104045                         ERROR   45              ;DID NOT TIME OUT OVER UNIBUS
020622  012737  020540  001110  10$:    MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
020630  005037  001264                 CLR     CPUEXP          ;NO CPU TRAPS EXPECTED IN NEXT TEST

                                        ;**************************************************************
                                        ;*TEST 37       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.
                                        ;*
                                        ;*      THIS TEST IS BEING RUN WITH ALL MEMORY REFERENCES GOING THRU
                                        ;*      THE UNIBUS MAP.
                                        ;*
                                        ;**************************************************************
020634                                  TST37:
```

```
                                        SCOPE
020634  000004                          MOV     #TST40,NXTTST   ;SAVE STARTING ADDRESS OF NEXT TEST
020636  012737  021550  001324                                  ;FOR ESCAPE ON PARITY ERRORS
                                        MOV     #060000,@#060000  ;MAKE SURE THAT ADDRESS 060000
020644  012737  060000  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
                                ;;
020652  012737  020704  001110  100$:  MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
020660  005077  160362                  CLR     @LREGU          ;CLEAR UPPER BITS OF MAPPING REG
020664  012777  060000  160352          MOV     #060000,@LREGL  ;LOAD LOWER BITS OF MAPPING REG
020672  013737  001240  172354          MOV     LOWEST,@#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
020700  012700  140000                  MOV     #140000,R0      ;SELECT PAR6, OFFSET IS 00000
020704  000240                  1$:     NOP                     ;THIS IS A SYNC POINT FOR SCOPING
020706  011001                          MOV     (R0),R1         ;READ LOCATION 060000 THRU THE UNIBUS
020710  012702  060000                  MOV     #060000,R2      ;THE EXPECTED PHYSICAL ADDRESS IS 060000
020714  020102                          CMP     R1,R2           ;SEE IF THE MAP'S FETCH WAS CORRECT
020716  001401                          BEQ     2$              ;BRANCH IF FETCHED DATA MATCHES ADDRESS
020720  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
                                ;;
020722  012737  020754  001110  2$:   MOV     #3$,$LPERR      ;SET LOOP ON ERROR POINTER TO 3$
020730  005077  160312                  CLR     @LREGU          ;CLEAR UPPER BITS OF MAPPING REG
020734  012777  052524  160302          MOV     #052524,@LREGL  ;LOAD LOWER BITS OF MAPPING REG
020742  013737  001240  172354          MOV     LOWEST,@#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
020750  012700  145252                  MOV     #145252,R0      ;SELECT PAR6, OFFSET IS 05252
020754  000240                  3$:     NOP                     ;THIS IS A SYNC POINT FOR SCOPING
020756  011001                          MOV     (R0),R1         ;READ LOCATION 057776 THRU THE UNIBUS
020760  012702  057776                  MOV     #057776,R2      ;THE EXPECTED PHYSICAL ADDRESS IS 057776
020764  020102                          CMP     R1,R2           ;SEE IF THE MAP'S FETCH WAS CORRECT
020766  001401                          BEQ     4$              ;BRANCH IF FETCHED DATA MATCHES ADDRESS
020770  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
                                ;;
020772  012737  021024  001110  4$:   MOV     #5$,$LPERR      ;SET LOOP ON ERROR POINTER TO 5$
021000  005077  160242                  CLR     @LREGU          ;CLEAR UPPER BITS OF MAPPING REG
021004  012777  045252  160232          MOV     #045252,@LREGL  ;LOAD LOWER BITS OF MAPPING REG
021012  013737  001240  172354          MOV     LOWEST,@#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
021020  012700  152524                  MOV     #152524,R0      ;SELECT PAR6, OFFSET IS 12524
021024  000240                  5$:     NOP                     ;THIS IS A SYNC POINT FOR SCOPING
021026  011001                          MOV     (R0),R1         ;READ LOCATION 057776 THRU THE UNIBUS
021030  012702  057776                  MOV     #057776,R2      ;THE EXPECTED PHYSICAL ADDRESS IS 057776
021034  020102                          CMP     R1,R2           ;SEE IF THE MAP'S FETCH WAS CORRECT
021036  001401                          BEQ     6$              ;BRANCH IF FETCHED DATA MATCHES ADDRESS
021040  104046                          ERROR   46              ;FAULTY RELOCATION BY UNIBUS MAP
                                ;;
                                ;*THE RELOCATION HERE USES A BASE OF 00050420 AND AN
                                ;*OFFSET OF 10420 TO PRODUCE AN ADDRESS OF 00061040
                                ;;
021042  012737  021074  001110  6$:   MOV     #7$,$LPERR      ;SET LOOP ON ERROR POINTER TO 7$
021050  005077  160172                  CLR     @LREGU          ;CLEAR UPPER BITS OF MAPPING REG
```

```
4037  021054  012777  050420  160162          MOV     #050420,@LREGL  ;LOAD LOWER BITS OF MAPPING REG
4038  021062  013737  001240  172354          MOV     LOWEST,@#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
4039  021070  012700  150420                  MOV     #150420,R0      ;SELECT PAR6, OFFSET IS 10420
4040  021074  000240                  7$:     NOP                     ;THIS IS A SYNC POINT FOR SCOPING
4041  021076  011001                          MOV     (R0),R1         ;READ LOCATION 061040 THRU THE UNIBUS
4042  021100  012702  061040                  MOV     #061040,R2      ;THE EXPECTED PHYSICAL ADDRESS IS 061040
4043  021104  020102                          CMP     R1,R2           ;SEE IF THE MAP'S FETCH WAS CORRECT
4044  021106  001401                          BEQ     8$              ;BRANCH IF FETCHED DATA MATCHES ADDRESS
4045  021110  104046                          ERROR   46              ;FAULTY RELOCATION BY UNIBUS MAP
4046                                  ;;
4047                                  ;*THE RELOCATION HERE USES A BASE OF 00054630 AND AN
4048                                  ;*OFFSET OF 04210 TO PRODUCE AN ADDRESS OF 00061040
4049                                  ;;
4050  021112  012737  021144  001110  8$:     MOV     #9$,$LPERR      ;SET LOOP ON ERROR POINTER TO 9$
4051  021120  005077  160122                  CLR     @LREGU          ;CLEAR UPPER BITS OF MAPPING REG
4052  021124  012777  054630  160112          MOV     #054630,@LREGL  ;LOAD LOWER BITS OF MAPPING REG
4053  021132  013737  001240  172354          MOV     LOWEST,@#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
4054  021140  012700  144210                  MOV     #144210,R0      ;SELECT PAR6, OFFSET IS 04210
4055  021144  000240                  9$:     NOP                     ;THIS IS A SYNC POINT FOR SCOPING
4056  021146  011001                          MOV     (R0),R1         ;READ LOCATION 061040 THRU THE UNIBUS
4057  021150  012702  061040                  MOV     #061040,R2      ;THE EXPECTED PHYSICAL ADDRESS IS 061040
4058  021154  020102                          CMP     R1,R2           ;SEE IF THE MAP'S FETCH WAS CORRECT
4059  021156  001401                          BEQ     10$             ;BRANCH IF FETCHED DATA MATCHES ADDRESS
4060  021160  104046                          ERROR   46              ;FAULTY RELOCATION BY UNIBUS MAP
4061                                  ;;
4062                                  ;*THE RELOCATION HERE USES A BASE OF 00044210 AND AN
4063                                  ;*OFFSET OF 14630 TO PRODUCE AN ADDRESS OF 00061040
4064                                  ;;
4065  021162  012737  021214  001110  10$:    MOV     #11$,$LPERR     ;SET LOOP ON ERROR POINTER TO 11$
4066  021170  005077  160052                  CLR     @LREGU          ;CLEAR UPPER BITS OF MAPPING REG
4067  021174  012777  044210  160042          MOV     #044210,@LREGL  ;LOAD LOWER BITS OF MAPPING REG
4068  021202  013737  001240  172354          MOV     LOWEST,@#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
4069  021210  012700  154630                  MOV     #154630,R0      ;SELECT PAR6, OFFSET IS 14630
4070  021214  000240                  11$:    NOP                     ;THIS IS A SYNC POINT FOR SCOPING
4071  021216  011001                          MOV     (R0),R1         ;READ LOCATION 061040 THRU THE UNIBUS
4072  021220  012702  061040                  MOV     #061040,R2      ;THE EXPECTED PHYSICAL ADDRESS IS 061040
4073  021224  020102                          CMP     R1,R2           ;SEE IF THE MAP'S FETCH WAS CORRECT
4074  021226  001401                          BEQ     12$             ;BRANCH IF FETCHED DATA MATCHES ADDRESS
4075  021230  104046                          ERROR   46              ;FAULTY RELOCATION BY UNIBUS MAP
4076                                  ;;
4077                                  ;*THE RELOCATION HERE USES A BASE OF 00056734 AND AN
4078                                  ;*OFFSET OF 02104 TO PRODUCE AN ADDRESS OF 00061040
4079                                  ;;
4080  021232  012737  021264  001110  12$:    MOV     #13$,$LPERR     ;SET LOOP ON ERROR POINTER TO 13$
4081  021240  005077  160002                  CLR     @LREGU          ;CLEAR UPPER BITS OF MAPPING REG
4082  021244  012777  056734  157772          MOV     #056734,@LREGL  ;LOAD LOWER BITS OF MAPPING REG
4083  021252  013737  001240  172354          MOV     LOWEST,@#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
4084  021260  012700  142104                  MOV     #142104,R0      ;SELECT PAR6, OFFSET IS 02104
4085  021264  000240                  13$:    NOP                     ;THIS IS A SYNC POINT FOR SCOPING
4086  021266  011001                          MOV     (R0),R1         ;READ LOCATION 061040 THRU THE UNIBUS
4087  021270  012702  061040                  MOV     #061040,R2      ;THE EXPECTED PHYSICAL ADDRESS IS 061040
4088  021274  020102                          CMP     R1,R2           ;SEE IF THE MAP'S FETCH WAS CORRECT
4089  021276  001401                          BEQ     14$             ;BRANCH IF FETCHED DATA MATCHES ADDRESS
4090  021300  104046                          ERROR   46              ;FAULTY RELOCATION BY UNIBUS MAP
4091                                  ;;
4092                                  ;*THE RELOCATION HERE USES A BASE OF 00042104 AND AN
```

```
4093                                        ;*OFFSET OF 16734 TO PRODUCE AN ADDRESS OF 00061040
4094                                        ;
4095  021302  012737  021334  001110  14$:   MOV     #15$,$LPERR     ;SET LOOP ON ERROR POINTER TO 15$
4096  021310  005077  157732                CLR     @LREGU          ;CLEAR UPPER BITS OF MAPPING REG
4097  021314  012777  042104  157722        MOV     #042104,@LREGL  ;LOAD LOWER BITS OF MAPPING REG
4098  021322  013737  001240  172354        MOV     LOWEST,@#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
4099  021330  012700  156734                MOV     #156734,R0      ;SELECT PAR6, OFFSET IS 16734
4100  021334  000240              15$:      NOP                     ;THIS IS A SYNC POINT FOR SCOPING
4101  021336  011001                        MOV     (R0),R1         ;READ LOCATION 061040 THRU THE UNIBUS
4102  021340  012702  061040                MOV     #061040,R2      ;THE EXPECTED PHYSICAL ADDRESS IS 061040
4103  021344  020102                        CMP     R1,R2           ;SEE IF THE MAP'S FETCH WAS CORRECT
4104  021346  001401                        BEQ     16$             ;BRANCH IF FETCHED DATA MATCHES ADDRESS
4105  021350  104046                        ERROR   46              ;FAULTY RELOCATION BY UNIBUS MAP
4106                                        ;
4107                                        ;*THE RELOCATION HERE USES A BASE OF 00057776 AND AN
4108                                        ;*OFFSET OF 01042 TO PRODUCE AN ADDRESS OF 00061040
4109                                        ;
4110  021352  012737  021404  001110  16$:   MOV     #17$,$LPERR     ;SET LOOP ON ERROR POINTER TO 17$
4111  021360  005077  157662                CLR     @LREGU          ;CLEAR UPPER BITS OF MAPPING REG
4112  021364  012777  057776  157652        MOV     #057776,@LREGL  ;LOAD LOWER BITS OF MAPPING REG
4113  021372  013737  001240  172354        MOV     LOWEST,@#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
4114  021400  012700  141042                MOV     #141042,R0      ;SELECT PAR6, OFFSET IS 01042
4115  021404  000240              17$:      NOP                     ;THIS IS A SYNC POINT FOR SCOPING
4116  021406  011001                        MOV     (R0),R1         ;READ LOCATION 061040 THRU THE UNIBUS
4117  021410  012702  061040                MOV     #061040,R2      ;THE EXPECTED PHYSICAL ADDRESS IS 061040
4118  021414  020102                        CMP     R1,R2           ;SEE IF THE MAP'S FETCH WAS CORRECT
4119  021416  001401                        BEQ     18$             ;BRANCH IF FETCHED DATA MATCHES ADDRESS
4120  021420  104046                        ERROR   46              ;FAULTY RELOCATION BY UNIBUS MAP
4121                                        ;
4122                                        ;*THE RELOCATION HERE USES A BASE OF 00041042 AND AN
4123                                        ;*OFFSET OF 17776 TO PRODUCE AN ADDRESS OF 00061040
4124                                        ;
4125  021422  012737  021454  001110  18$:   MOV     #19$,$LPERR     ;SET LOOP ON ERROR POINTER TO 19$
4126  021430  005077  157612                CLR     @LREGU          ;CLEAR UPPER BITS OF MAPPING REG
4127  021434  012777  041042  157602        MOV     #041042,@LREGL  ;LOAD LOWER BITS OF MAPPING REG
4128  021442  013737  001240  172354        MOV     LOWEST,@#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
4129  021450  012700  157776                MOV     #157776,R0      ;SELECT PAR6, OFFSET IS 17776
4130  021454  000240              19$:      NOP                     ;THIS IS A SYNC POINT FOR SCOPING
4131  021456  011001                        MOV     (R0),R1         ;READ LOCATION 061040 THRU THE UNIBUS
4132  021460  012702  061040                MOV     #061040,R2      ;THE EXPECTED PHYSICAL ADDRESS IS 061040
4133  021464  020102                        CMP     R1,R2           ;SEE IF THE MAP'S FETCH WAS CORRECT
4134  021466  001401                        BEQ     20$             ;BRANCH IF FETCHED DATA MATCHES ADDRESS
4135  021470  104046                        ERROR   46              ;FAULTY RELOCATION BY UNIBUS MAP
4136                                        ;
4137                                        ;*THE RELOCATION HERE USES A BASE OF 00057776 AND AN
4138                                        ;*OFFSET OF 00002 TO PRODUCE AN ADDRESS OF 00060000
4139                                        ;
4140  021472  012737  021524  001110  20$:   MOV     #21$,$LPERR     ;SET LOOP ON ERROR POINTER TO 21$
4141  021500  005077  157542                CLR     @LREGU          ;CLEAR UPPER BITS OF MAPPING REG
4142  021504  012777  057776  157532        MOV     #057776,@LREGL  ;LOAD LOWER BITS OF MAPPING REG
4143  021512  013737  001240  172354        MOV     LOWEST,@#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
4144  021520  012700  140002                MOV     #140002,R0      ;SELECT PAR6, OFFSET IS 00002
4145  021524  000240              21$:      NOP                     ;THIS IS A SYNC POINT FOR SCOPING
4146  021526  011001                        MOV     (R0),R1         ;READ LOCATION 060000 THRU THE UNIBUS
4147  021530  012702  060000                MOV     #060000,R2      ;THE EXPECTED PHYSICAL ADDRESS IS 060000
4148  021534  020102                        CMP     R1,R2           ;SEE IF THE MAP'S FETCH WAS CORRECT
```

```
                                                BEQ     22$             ;BRANCH IF FETCHED DATA MATCHES ADDRESS
                                                ERROR   46              ;FAULTY RELOCATION BY UNIBUS MAP
021542  012737  020652  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.
                                        ;*
                                        ;;*****************************************************************
021550                                  TST40:
                                                SCOPE
021550  000004
021552  012737  022054  001324                 MOV     #TST41,NXTTST   ;SAVE STARTING ADDRESS OF NEXT TEST
                                                                        ;FOR ESCAPE ON PARITY ERRORS
021560  012737  000012  001204                 MOV     #12,$TIMES      ;;DO 12 ITERATIONS
021566  012737  177777  021722  20$:          MOV     #-1,4$          ;INITIALIZE FLAG AS NEGATIVE ONE
021574  005077  157446                         CLR     @LREGU          ;CLEAR UPPER 6 BITS OF MAP REG
021600  012777  020000  157436                 MOV     #20000,@LREGL   ;LOAD 4K BASE INTO MAP REGISTER
021606  012701  100100                         MOV     #100100,R1      ;LOAD BITS TO SELECT PAR 4, OFFSET 100
021612  012700  150000                         MOV     #150000,R0      ;LOAD BITS TO SELECT PAR 6, OFFSET 2K
021616  012737  000277  172350                 MOV     #277,KIPAR4     ;START WITH PHYSICAL 6K
021624  013737  001240  172354                 MOV     LOWEST,KIPAR6   ;LOAD PAR 6 WITH MAP REG'S ADDR
021632  012737  021672  001110                 MOV     #10$,$LPERR     ;SET LOOP ON ERROR POINTER TO 10$
021640  012737  000020  001264  3$:           MOV     #TIMOUT,CPUEXP  ;EXPECTING A UNIBUS TIME OUT DURING TEST
021646  005037  001266                         CLR     PCPUER          ;CLEAR TIME OUT FLAG
021652  013710  001326                         MOV     DATA,(R0)       ;THIS LOAD WILL TIME OUT WHEN YOU
                                                                        ;HAVE REACHED THE TOP OF MEMORY
                                                                        ;IT SELECTS PAR 6 WHICH WILL PUT ADDR
                                                                        ;<XXX XX1>0000 ON THE UNIBUS.
                                                                        ;THE X'S WILL SELECT THE LOWEST USEABLE
                                                                        ;MAPPING REGISTER.  THE DEFAULT CASE IS
                                                                        ;010000,  SELECTING MAPPING REGISTER 0.
021656  005737  001266                         TST     PCPUER          ;SEE IF THERE WAS MAIN MEMORY
021662  001016                                 BNE     1$              ;BRANCH IF NO MAIN MEMORY FROM UNIBUS
021664  012737  000040  001264                 MOV     #NEXMEM,CPUEXP  ;POSSIBLE CACHE NON-EXISTANT MEMORY
021672  011103                         10$:    MOV     (R1),R3         ;READ TEST LOCATION VIA FASTBUS
021674  022737  000040  001266                 CMP     #NEXMEM,PCPUER  ;WAS THERE CACHE NON-EXISTANT MEMORY
021702  001414                                 BEQ     2$              ;BRANCH IF NON-EXISTANT MEMORY
021704  000240                                 NOP                     ;THIS IS A SYNC POINT FOR SCOPING
021706  011002                                 MOV     (R0),R2         ;READ TEST LOCATION VIA UNIBUS MAP
021710  020203                                 CMP     R2,R3           ;COMPARE TEST DATA R2=MAP DATA
                                                                        ;R3=FASTBUS DATA
021712  001410                                 BEQ     2$              ;BRANCH IF IT WAS THE SAME
021714  104047                                 ERROR   47              ;FAILURE IN CARRY PROPAGATION IN MAP
021716  000406                                 BR      2$              ;BRANCH TO UPDATE ROUTINE
021720  005227                         1$:     INC     (PC)+           ;INCREMENT ONE TIME ENTRANCE FLAG
021722  177777                         4$:     .WORD   -1              ;USE NEGATIVE ONE FOR FLAG
021724  001003                                 BNE     2$              ;BRANCH IF YOU'VE BEEN HERE BEFORE
021726  013737  172350  001320                 MOV     KIPAR4,RSIZE    ;SAVE UPPER LIMIT OF MEMORY
```

```
4205  021734  062737  000100  001326  2$:     ADD     #100,DATA       ;CHANGE PATTERN FOR NEXT LOAD
4206  021742  062737  000100  172350          ADD     #100,KIPAR4     ;ADD 2K TO PAR4
4207  021750  062777  010000  157266          ADD     #10000,@LREGL   ;ADD 2K TO MAP REGISTER
4208  021756  001330                          BNE     3$              ;BRANCH IF MAP REGISTER NOT ZERO
4209  021760  005277  157262                  INC     @LREGU          ;ADD ONE TO UPPER 6 BITS OF MAP REG
4210  021764  022777  000073  157254          CMP     #73,@LREGU      ;SEE IF TOP 128K BLOCK HAS BEEN PASSED
4211  021772  103322                          BHIS    3$              ;BRANCH IF NOT PAST IT
4212  021774  005237  001326                  INC     DATA            ;CHANGE DATA PATTERN FOR NEXT PASS
4213  022000  042737  177700  001326          BIC     #177700,DATA    ;CLEAR UPPER 10 BITS OF DATA PATTERN
4214  022006  052737  000300  001326          BIS     #300,DATA       ;START WITH 3XX IN DATA PATTERN
4215  022014  012737  021566  001110          MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
4216  022022  023737  177760  001320          CMP     @#SIZELO,RSIZE  ;SEE IF SIZE JUMPERS AGREE WITH MEMORY TOP
4217  022030  001407                          BEQ     19$             ;BRANCH IF TOP OF MEMORY AGREES WITH
                                                                      ;THE SIZE JUMPERS
4218
4219  022032  105737  001331                  TSTB    KB11EM          ;RUNNING ON A KB11-EM?
4220  022036  001004                          BNE     19$             ;SKIP ERROR IF YES
4221  022040  105737  001332                  TSTB    KB11CM          ;RUNNING ON MODIFIED MACHINE?
4222  022044  001001                          BNE     19$             ;BRANCH IF YES
4223  022046  104050                          ERROR   50              ;MEMORY SIZE JUMPERS NOT RIGHT
4224  022050  005037  001264          19$:    CLR     CPUEXP          ;NO CPU TRAPS EXPECTED IN NEXT TEST
4225
4226
4227
4228                                  ;*****************************************************************
4229                                  ;*TEST 41       PARITY REPORTING THRU THE MAP, MAIN MEMORY EVEN WORD
4230                                  ;*
4231                                  ;*      THIS TEST FORCES A PARITY ERROR IN MAIN MEMORY AT PHYSICAL
4232                                  ;*      ADDRESS 00040000 BY SETTING ONE BIT IN EACH BYTE OF THE
4233                                  ;*      WORD.  THE TEST THEN FORCES THE EVEN WORD PARITY BITS TO
4234                                  ;*      ONES AND READS ADDRESS 00040000 THRU THE UNIBUS MAP.  THEN
4235                                  ;*      IT CHECKS TO SEE THAT THE PARITY ABORT OCCURRED AND THAT THE
4236                                  ;*      CONDITION WAS CORRECT BEFORE GOING TO THE NEXT TEST.
4237                                  ;*
4238                                  ;*      ERRORS ARE REPORTED INLINE IN THE TEST CODE, AND THE PARITY
4239                                  ;*      VECTOR WILL TRAP TO LABEL 10$ AT THE END OF THIS TEST.
4240                                  ;*
4241                                  ;*****************************************************************
4242  022054                  TST41:
4243  022054  000004                          SCOPE
4244  022056  012737  022316  001324          MOV     #TST42,NXTTST   ;SAVE STARTING ADDRESS OF NEXT TEST
4245                                                                  ;FOR ESCAPE ON PARITY ERRORS
4246  022064  104416                          TBITO                   ;TURN OFF T BIT TRAPPING IF ON
4247  022066  012737  023404  001200  20$:    MOV     #23404,$TMP4    ;EXPECTED ERROR CONDITION
4248  022074  005037  001274                  CLR     PPARER          ;CLEAR MEM ERROR REG STORAGE
4249  022100  012737  177777  177744          MOV     #-1,@#MEMERR    ;CLEAR MEMORY ERROR REGISTER
4250  022106  013737  177746  001202          MOV     @#CONTRL,$TMP5  ;SAVE CONTROL REG TO RESTORE CACHE
4251  022114  012737  000014  177746          MOV     #14,@#CONTRL    ;FORCE MISSES IN CACHE
4252  022122  013737  001240  172354          MOV     LOWEST,KIPAR6   ;PUT UNIBUS ADDR OF MAP REG IN PAR6
4253  022130  012777  040000  157106          MOV     #40000,@LREGL   ;LOWEST MAP REGISTER POINTS TO 8K
4254  022136  005077  157104                  CLR     @LREGU          ;CLEAR UPPER BITS OF MAP REG
4255  022142  012701  177750                  MOV     #MAINT,R1       ;PUT ADDR OF MAINT REG IN R1
4256  022146  012737  022164  001110          MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
4257  022154  012700  140000                  MOV     #140000,R0      ;SELECT KIPAR6, OFFSET
4258  022160  012710  020001                  MOV     #20001,(R0)     ;LOAD ONE BIT IN EACH BYTE OF 40000
4259  022164  012737  022256  000114  1$:     MOV     #10$,CACHVEC    ;SET PARITY VECTOR TO 10$
4260  022172  012737  030000  177750          MOV     #30000,@#MAINT  ;FORCE EVEN WORD PARITY BITS TO ONE
```

```
4261  022200  000240                          NOP                     ;THIS IS A SYNC POINT FOR SCOPING
4262  022202  011003                          MOV     (R0),R3         ;'DATA FETCH'' SHOULD CAUSE PARITY ABORT
4263  022204  011102                          MOV     (R1),R2         ;SAVE MAINT REG IN CASE NO TRAP
4264  022206  005011                          CLR     (R1)            ;CLEAR MAINT REG IN CASE NO TRAP
4265  022210  050000                          BIS     R0,R0           ;DUMMY INST WITH P.B.'S ON
4266  022212  023737  001200  001274          CMP     $TMP4,PPARER    ;SEE IF ERROR REG HOLDS RIGHT DATA
4267  022220  001401                          BEQ     2$              ;BRANCH IF CONDITION WAS CORRECT
4268  022222  104051                          ERROR   51              ;FAILURE IN PARITY REPORTING
4269  022224  012737  177777  177744  2$:     MOV     #-1,MEMERR      ;CLEAR MEMORY ERROR REGISTER
4270  022232  012737  005764  000114          MOV     #MEMER,CACHVEC  ;RESTORE PARITY SERVICE ROUTINE
4271  022240  013737  001202  177746          MOV     $TMP5,CNTRL     ;RESTORE CACHE TO FORMER CONDITION
4272  022246  012737  022066  001110          MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
4273  022254  000420                          BR      TST42           ;;TEST OVER GO TO NEXT TEST
4274
4275
4276  022256  013737  177740  001270  10$:     MOV     @#LOADRS,PLOADR ;SAVE LOWER CACHE ADDR REG
4277  022264  013737  177742  001272          MOV     @#HIADRS,PHIADR ;SAVE HIGH BITS OF FAILING ADDR
4278  022272  013737  177744  001274          MOV     @#MEMERR,PPARER ;SAVE MEMORY ERROR REGISTER
4279  022300  013737  177746  001276          MOV     @#CNTRL,PCONTR  ;SAVE CONTROL REGISTER FOR TYPE OUT
4280  022306  012737  030000  001300          MOV     #30000,PMAINT   ;SAVE DATA IN MAINTENENCE REGISTER
4281  022314  000002                          RTI                     ;RETURN TO TEST AND CHECK PARITY
4282
4283
4284                          ;*****************************************************************
4285                          ;*TEST 42       PARITY REPORTING THRU THE MAP, MAIN MEMORY ODD WORD
4286                          ;*
4287                          ;*      THIS TEST FORCES A PARITY ERROR IN MAIN MEMORY AT PHYSICAL
4288                          ;*      ADDRESS 00040002 BY SETTING ONE BIT IN EACH BYTE OF THE
4289                          ;*      WORD.  THE TEST THEN FORCES THE ODD WORD PARITY BITS TO
4290                          ;*      ONES AND READS ADDRESS 00040002 THRU THE UNIBUS MAP.  THEN
4291                          ;*      IT CHECKS TO SEE THAT THE PARITY ABORT OCCURRED AND THAT THE
4292                          ;*      CONDITION WAS CORRECT BEFORE GOING TO THE NEXT TEST.
4293                          ;*
4294                          ;*      ERRORS ARE REPORTED INLINE IN THE TEST CODE, AND THE PARITY
4295                          ;*      VECTOR WILL TRAP TO LABEL 10$ AT THE END OF THIS TEST.
4296                          ;*
4297                          ;*      THIS TEST IS BEING RUN WITH ALL MEMORY REFERENCES GOING THRU
4298                          ;*      THE UNIBUS MAP.
4299                          ;*
4300                          ;*****************************************************************
4301  022316                  TST42:
4302  022316  000004                          SCOPE
4303  022320  012737  022570  001324          MOV     #TST43,NXTTST   ;SAVE STARTING ADDRESS OF NEXT TEST
4304                                                                  ;FOR ESCAPE ON PARITY ERRORS
4305  022326  012737  023410  001200  20$:     MOV     #23410,$TMP4    ;LOAD EXPECTED ERROR CONDITION
4306  022334  005037  001274                  CLR     PPARER          ;CLEAR MEM ERROR REG STORAGE
4307  022340  012737  177777  177744          MOV     #-1,@#MEMERR    ;CLEAR MEMORY ERROR REGISTER
4308  022346  013737  177746  001202          MOV     @#CNTRL,$TMP5   ;SAVE CONTROL REG TO RESTORE CACHE
4309  022354  012737  000014  177746          MOV     #14,@#CNTRL     ;FORCE MISSES IN CACHE
4310  022362  013737  001240  172354          MOV     LOWEST,KIPAR6   ;PUT UNIBUS ADDR OF MAP REG IN PAR6
4311  022370  012777  040000  156646          MOV     #40000,@LREGL   ;LOWEST MAP REGISTER POINTS TO 8K
4312  022376  005077  156644                  CLR     @LREGU          ;CLEAR UPPER BITS OF MAP REG
4313  022402  012701  177750                  MOV     #MAINT,R1       ;PUT ADDR OF MAINT REG IN R1
4314  022406  012737  022436  001110          MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
4315  022414  005037  140000                  CLR     140000          ;CLEAR LOCATION 40000
4316  022420  012700  140002                  MOV     #140002,R0      ;LOAD ADDRESS OF 40002 INTO R0
```

```
4317  022424  012737  022436  001110          MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
4318  022432  012710  020001                  MOV     #20001,(R0)     ;LOAD ONE BIT IN EACH BYTE OF 40002
4319  022436  012737  022530  000114  1$:     MOV     #10$,CACHVEC    ;SET PARITY VECTOR TO 10$
4320  022444  012737  140000  177750          MOV     #140000,@#MAINT ;FORCE ODD WORD PARITY BITS TO ONE
4321  022452  000240                          NOP                     ;THIS IS A SYNC POINT FOR SCOPING
4322  022454  011003                          MOV     (R0),R3         ;''DATA FETCH'' SHOULD CAUSE PARITY ABORT
4323  022456  011102                          MOV     (R1),R2         ;SAVE MAINT REG IN CASE NO TRAP
4324  022460  005011                          CLR     (R1)            ;CLEAR MAINT REG IN CASE NO TRAP
4325  022462  050000                          BIS     R0,R0           ;DUMMY INST WITH P.B.'S ON
4326  022464  023737  001200  001274          CMP     $TMP4,PPARER    ;SEE IF MEM ERROR REG HOLDS RIGHT DATA
4327  022472  001401                          BEQ     2$              ;BRANCH IF ERROR WAS EXPECTED ONE
4328  022474  104051                          ERROR   51              ;FAILURE IN PARITY REPORTING
4329  022476  012737  177777  177744  2$:     MOV     #-1,MEMERR      ;CLEAR MEMORY ERROR REGISTER
4330  022504  012737  005764  000114          MOV     #MEMER,CACHVEC  ;RESTORE PARITY SERVICE ROUTINE
4331  022512  013737  001202  177746          MOV     $TMP5,CONTRL    ;RESTORE CACHE TO FORMER CONDITION
4332  022520  012737  022326  001110          MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
4333  022526  000420                          BR      TST43           ;;TEST IS OVER, GO TO NEXT TEST
4334
4335
4336  022530  013737  177740  001270  10$:     MOV     @#LOADRS,PLOADR ;SAVE LOWER CACHE ADDR REG
4337  022536  013737  177742  001272          MOV     @#HIADRS,PHIADR ;SAVE HIGH BITS OF FAILING ADDR
4338  022544  013737  177744  001274          MOV     @#MEMERR,PPARER ;SAVE MEMORY ERROR REGISTER
4339  022552  013737  177746  001276          MOV     @#CONTRL,PCONTR ;SAVE CONTROL REGISTER FOR TYPE OUT
4340  022560  012737  140000  001300          MOV     #140000,PMAINT  ;SAVE DATA IN MAINTENENCE REGISTER
4341  022566  000002                          RTI                     ;RETURN TO TEST AND CHECK PARITY
4342
4343                                  ;;**************************************************************
4344                                  ;;*TEST 43       PARITY REPORTING THRU THE MAP, CACHE DATA GROUP 0
4345                                  ;;*
4346                                  ;;*     THIS TEST FORCES A PARITY ERROR IN CACHE GROUP 0 AT PHYSICAL
4347                                  ;;*     ADDRESS 00040000 BY SETTING BIT 15 IN THE WORD.  THE TEST
4348                                  ;;*     THEN FORCES GROUP 0 LOW BYTE PARITY BIT TO ZERO AND EVEN
4349                                  ;;*     WORD HIGH BYTE PARITY BIT TO ONE BEFORE READING ADDRESS 00040000
4350                                  ;;*     THRU THE UNIBUS MAP.  THEN IT CHECKS TO SEE THAT THE PARITY
4351                                  ;;*     ABORT OCCURRED AND THAT THE CONDITION WAS CORRECT BEFORE
4352                                  ;;*     GOING TO THE NEXT TEST.
4353                                  ;;*
4354                                  ;;*     ERRORS ARE REPORTED INLINE IN THE TEST CODE, AND THE PARITY
4355                                  ;;*     VECTOR WILL TRAP TO LABEL 10$ AT THE END OF THIS TEST.
4356                                  ;;*
4357                                  ;;*     THIS TEST IS BEING RUN WITH ALL MEMORY REFERENCES GOING THRU
4358                                  ;;*     THE UNIBUS MAP.
4359                                  ;;*
4360                                  ;;**************************************************************
4361
4362  022570                          TST43:
4363  022570  000004                          SCOPE
4364  022572  012737  023056  001324          MOV     #TST44,NXTTST   ;SAVE STARTING ADDRESS OF NEXT TEST
4365                                                                  ;FOR ESCAPE ON PARITY ERRORS
4366  022600  012737  177777  022744  20$:     MOV     #-1,21$         ;RESTORE NEG ONE FOR NEXT ITERATION
4367  022606  032737  000004  177746          BIT     #BIT2,@#CONTRL  ;SEE IF GROUP 0 IS DISABLED
4368  022614  001120                          BNE     TST44           ;;BRANCH IF GROUP 0 IS DISABLED
4369  022616  013737  177746  001202          MOV     CONTRL,$TMP5    ;SAVE CONDITION OF CACHE
4370  022624  012737  023504  001200          MOV     #23504,$TMP4    ;EXPECTED PARITY CONDITION GROUP 0
4371  022632  005037  001274                  CLR     PPARER          ;CLEAR MEM ERROR REG STORAGE
4372  022636  012737  177777  177744          MOV     #-1,@#MEMERR    ;CLEAR MEMORY ERROR REGISTER
```

```
4373  022644  013737  001240  172354          MOV     LOWEST,KIPAR6  ;PUT UNIBUS ADDR OF MAP REG IN PAR6
4374  022652  005077  156370                  CLR     @LREGU         ;CLEAR UPPER BITS ON MAP REG
4375  022656  012777  040000  156360          MOV     #40000,@LREGL  ;LOAD LOWER 16 BITS OF MAP REG
4376  022664  012705  177750                  MOV     #MAINT,R5      ;PUT ADDR OF MAINT REG IN R5
4377  022670  012704  140000                  MOV     #140000,R4     ;SELECT PAR 6 BASE 0
4378  022674  012714  100000                  MOV     #BIT15,(R4)    ;MAKE SURE I GET SOFT AND HARD ERRORS
4379                                                                 ;BY LOADING 40000 WITH BIT 15, THIS
4380                                                                 ;CAUSES THE HIGH BYTE P.B. TO BE ON
4381                                                                 ;AND THE LOW BYTE P.B. TO BE OFF
4382  022700  012737  000030  177746          MOV     #30,@#CONTRL   ;FORCE SELECTION OF GROUP 0 ON READ
4383  022706  011401                          MOV     (R4),R1        ;PUT (40000) & (40002) IN GROUP 0
4384  022710  005001                          CLR     R1             ;ZERO R1 FOR THE MOMENT
4385  022712  012737  023016  000114          MOV     #10$,CACHVEC   ;SET PARITY VECTOR TO 10$
4386  022720  010137  177750          1$:     MOV     R1,@#MAINT     ;FORCE GPO PARITY BITS TO 0 ON 2ND PASS
4387  022724  000241                          CLC                    ;THIS IS A SYNC POINT FOR SCOPING
4388                                                                 ;THE LOW BYTE HAS ITS P.B. OFF
4389  022726  011401                          MOV     (R4),R1        ;DATA FETCH SHOULD CAUSE PARITY ABORT
4390  022730  011503                          MOV     (R5),R3        ;SAVE MAINT REG IN CASE NO TRAP
4391  022732  105015                          CLRB    (R5)           ;CLEAR MAINT REG IN CASE NO TRAP
4392  022734  006701                          SXT     R1             ;DUMMY INST WITH P.B.'S OFF
4393  022736  012701  020020                  MOV     #020020,R1     ;FORCE GROUP 0 LOW BYTE PARITY BIT
4394                                                                 ;TO 0 AND MAIN MEMORY EVEN WORD HIGH
4395                                                                 ;BYTE PARITY BIT TO 1
4396  022742  005227                  INC     (PC)+          ;INCREMENT NEXT WORD TO ZERO ON FIRST PASS
4397  022744  177777          21$:    .WORD   -1             ;
4398  022746  001764                  BEQ     1$             ;BRANCH ON FIRST PASS ONLY
4399  022750  023737  001200  001274  CMP     $TMP4,PPARER   ;SEE IF ERROR CONDITION MATCHES EXPECTED
4400  022756  001402                  BEQ     2$             ;BRANCH IF CORRECT CONDITION
4401  022760  010302                  MOV     R3,R2          ;R2 HOLDS MAINT REG FOR ERROR TYPE OUT
4402  022762  104051                  ERROR   51             ;FAILURE IN PARITY REPORTING
4403  022764  012737  177777  17774'  2$:     MOV     #-1,MEMERR     ;CLEAR MEMORY ERROR REGISTER
4404  022772  012737  005764  000114          MOV     @#MEMER,CACHVEC ;RESTORE PARITY SERVICE ROUTINE
4405  023000  013737  001202  177746          MOV     $TMP5,CONTRL   ;RESTORE CACHE TO FORMER CONDITION
4406  023006  012737  022600  001110          MOV     #20$,$LPERR    ;SET LOOP POINTER TO START OF TEST
4407  023014  000420                          BR      TST44          ;;TEST OVER GO TO NEXT TEST
4408
4409
4410  023016  013737  177740  001270  10$:     MOV     @#LOADRS,PLOADR ;SAVE LOWER CACHE ADDR REG
4411  023024  013737  177742  001272          MOV     @#HIADRS,PHIADR ;SAVE HIGH BITS OF FAILING ADDR
4412  023032  013737  177744  001274          MOV     @#MEMERR,PPARER ;SAVE MEMORY ERROR REGISTER
4413  023040  013737  177746  001276          MOV     @#CONTRL,PCONTR ;SAVE CONTROL REGISTER FOR TYPE OUT
4414  023046  012737  030060  001300          MOV     #30060,PMAINT  ;SAVE DATA IN MAINTENENCE REGISTER
4415  023054  000002                          RTI                    ;RETURN TO TEST AND CHECK PARITY
4416
4417
4418                  ;;*****************************************************************
4419                  ;*TEST 44       PARITY REPORTING THRU THE MAP, CACHE DATA GROUP 1
4420                  ;*
4421                  ;*      THIS TEST FORCES A PARITY ERROR IN CACHE GROUP 1 AT PHYSICAL
4422                  ;*      ADDRESS 00040000 BY SETTING BIT 15 IN THE WORD.  THE TEST
4423                  ;*      THEN FORCES GROUP 1 LOW BYTE PARITY BIT TO ZERO AND EVEN
4424                  ;*      WORD HIGH BYTE PARITY BIT TO ONE BEFORE READING ADDRESS 00040000
4425                  ;*      THRU THE UNIBUS MAP.  THEN IT CHECKS TO SEE THAT THE PARITY
4426                  ;*      ABORT OCCURRED AND THAT THE CONDITION WAS CORRECT BEFORE
4427                  ;*      GOING TO THE NEXT TEST.
4428                  ;*
```

```
                        ;*
                        ;*      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.
                        ;*
                        ;*************************************************************************
023056  000004         TST44:  SCOPE
023060  012737  023344  001324          MOV     #TST45,NXTTST   ;SET UP ESCAPE POINTER IN CASE OF
                                                                ;RANDOM PARITY ABORTS
023066  012737  177777  023232  20$:    MOV     #-1,21$         ;RESTORE NEG ONE FOR NEXT ITERATION
023074  032737  000010  177746          BIT     #BIT3,CONTRL    ;SEE IF GROUP 1 IS DISABLED
023102  001120                         BNE     TST45           ;;BRANCH TO NEXT TEST IF GROUP 1 DISABLED
023104  013737  177746  001202          MOV     CONTRL,$TMP5    ;SAVE CONDITION OF CACHE
023112  012737  023604  001200          MOV     #23604,$TMP4    ;EXPECTED PARITY CONDITION GROUP 1
023120  005037  001274                 CLR     PPARER          ;CLEAR MEM ERROR REG STORAGE
023124  012737  177777  177744          MOV     #-1,@#MEMERR    ;CLEAR MEMORY ERROR REGISTER
023132  013737  001240  172354          MOV     LOWEST,KIPAR6   ;PUT UNIBUS ADDR OF MAP REG IN PAR6
023140  005077  156102                 CLR     @LREGU          ;CLEAR UPPER BITS ON MAP REG
023144  012777  040000  156072          MOV     #40000,@LREGL   ;LOAD LOWER 16 BITS OF MAP REG
023152  012705  177750                 MOV     #MAINT,R5       ;PUT ADDR OF MAINT REG IN R5
023156  012704  140000                 MOV     #140000,R4      ;SELECT PAR 6 BASE 0
023162  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
023166  012737  000044  177746          MOV     #44,@#CONTRL    ;FORCE SELECTION OF GROUP 0 ON READ
023174  011401                         MOV     (R4),R1         ;PUT (40000) & (40002) IN GROUP 1
023176  005001                         CLR     R1              ;CLEAR R1 FOR MOMENT
023200  012737  023304  000114          MOV     #10$,CACHVEC    ;SET PARITY VECTOR TO 10$
023206  010137  177750          1$:    MOV     R1,@#MAINT      ;FORCE GP1 PARITY BITS TO 0 ON 2ND PASS
023212  000241                         CLC                     ;THIS IS A SYNC POINT FOR SCOPING
                                                                ;THE LOW BYTE HAS ITS P.B. OFF
023214  011401                         MOV     (R4),R1         ;DATA FETCH SHOULD CAUSE PARITY ABORT
023216  011503                         MOV     (R5),R3         ;SAVE MAINT REG IN CASE NO TRAP
023220  105015                         CLRB    (R5)            ;CLEAR MAINT REG IN CASE NO TRAP
023222  006701                         SXT     R1              ;DUMMY INST WITH P.B.'S OFF
023224  012701  020100                 MOV     #020100,R1      ;FORCE GROUP 1 LOW BYTE PARITY BIT
                                                                ;TO 0 AND MAIN MEMORY EVEN WORD HIGH
                                                                ;BYTE PARITY BIT TO 1
023230  005227                         INC     (PC)+           ;INCREMENT NEXT WORD TO 0
023232  177777                 21$:    .WORD   -1              ;
023234  001764                         BEQ     1$              ;BRANCH ON FIRST PASS ONLY
023236  023737  001200  001274          CMP     $TMP4,PPARER    ;SEE IF ERROR CONDITION MATCHES EXPECTED
023244  001402                         BEQ     2$              ;BRANCH IF ERROR WAS EXPECTED ONE
023246  010302                         MOV     R3,R2           ;R2 HOLDS MAINT REG FOR ERROR TYPE OUT
023250  104051                         ERROR   51              ;FAILURE IN PARITY REPORTING
023252  012737  177777  177744  2$.    MOV     #-1,MEMERR      ;CLEAR MEMORY ERROR REGISTER
023260  012737  005764  000114          MOV     #MEMER,CACHVEC  ;RESTORE PARITY SERVICE ROUTINE
023266  013737  001202  177746          MOV     $TMP5,CONTRL    ;RESTORE CACHE TO FORMER CONDITION
023274  012737  023066  001110          MOV     #2?$,$LPERR     ;SET LOOP POINTER TO START OF TEST
023302  000420                         BR      ?               ;;TEST OVER GO TO NEXT TEST


023304  013737  177740  001270  10$:    MOV     @#LOADRS,PLOADR ;SAVE LOWER CACHE ADDR REG
023312  013737  177742  001272          MOV     @#HIADRS,PHIADR ;SAVE HIGH BITS OF FAILING ADDR
```

```
4485  023320  013737  177744  001274          MOV     @#MEMERR,PPARER ;SAVE MEMORY ERROR REGISTER
4486  023326  013737  177746  001276          MOV     @#CONTRL,PCONTR ;SAVE CONTROL REGISTER FOR TYPE OUT
4487  023334  012737  030300  001300          MOV     #30300,PMAINT   ;SAVE DATA IN MAINTENENCE REGISTER
4488  023342  000002                          RTI                     ;RETURN TO TEST AND CHECK PARITY
4489
4490
4491                                  ;;*************************************************************************
4492                                  ;*TEST 45       CHECK BYP BITS IN UBMR
4493                                  ;* THIS TEST IS EXECUTED ONLY ON KB11-EM OR 11/74          (KB11CM)
4494                                  ;*
4495                                  ;;*************************************************************************
4496  023344  000004                  TST45:  SCOPE
4497  023346  012737  023500  001324          MOV     #TST46,NXTTST
4498  023354  105737  001331                  TSTB    KB11EM          ;IS THIS A KB11-EM?
4499  023360  001003                          BNE     10$             ;BR IF YES
4500  023362  105737  001332                  TSTB    KB11CM          ;IS THIS A MODIFIED PROCESSOR?
4501  023366  001444                          BEQ     5$
4502  023370  012700  170202          10$:    MOV     #170202,R0
4503  023374  005010                  4$:     CLR     (R0)
4504  023376  032710  100000                  BIT     #BYP,(R0)
4505  023402  001405                          BEQ     1$
4506  023404  010037  001154                  MOV     R0,$REG0
4507  023410  011037  001156                  MOV     (R0),$REG1
4508  023414  104053                          ERROR   53
4509
4510  023416  052710  100000          1$:     BIS     #BYP,(R0)
4511  023422  032710  100000                  BIT     #BYP,(R0)
4512  023426  001005                          BNE     2$
4513  023430  010037  001154                  MOV     R0,$REG0
4514  023434  011037  001156                  MOV     (R0),$REG1
4515  023440  104054                          ERROR   54
4516
4517  023442  042710  100000          2$:     BIC     #BYP,(R0)
4518  023446  032710  100000                  BIT     #BYP,(R0)
4519  023452  001405                          BEQ     3$
4520  023454  010037  001154                  MOV     R0,$REG0
4521  023460  011037  001156                  MOV     (R0),$REG1
4522  023464  104053                          ERROR   53
4523
4524  023466  062700  000004          3$:     ADD     #4,R0
4525  023472  020027  170366                  CMP     R0,#170366
4526  023476  001336                          BNE     4$
4527  023500                          5$:
4528
4529
4530                                  ;;*************************************************************************
4531                                  ;*TEST 46       CHECK FOR PRESENCE OF MKA11 REGS. ON KB11-EM OR KB11CM
4532                                  ;* THIS TEST IS EXECUTED ONLY ON KB11-EM OR 11/74          (KB11CM)
4533                                  ;*
4534                                  ;;*************************************************************************
4535  023500  000004                  TST46:  SCOPE
4536  023502  012737  023640  001324          MOV     #TST47,NXTTST
4537          172100                  MKCSR=172100
4538
4539          177744                  MSER= 177744
4540  023510  105737  001331                  TSTB    KB11EM          ;KB11-EM?
```

```
4541  023514  001003                        BNE     1$                      ;BR IF YES
4542  023516  105737  001332                TSTB    KB11CM                  ;MODIFIED CPU?
4543  023522  001446                        BEQ     99$                     ;NO, SKIP THIS TEST
4544  023524  012702  172100        1$:     MOV     #MKCSR, R2              ;POINT TO CSR REGISTERS.
4545  023530  013746  000004                MOV     @#4, -(SP)              ;SAVE OLD CPU ERROR VEC
4546  023534  012737  023560  000004        MOV     #66$, @#4               ;REPLACE WITH TEST ROUTINE
4547
4548  023542  005712                2$:     TST     (R2)                    ;TRY TO ACCESS CSR
4549  023544  062702  000002                ADD     #2,R2                   ;INCREMENT R2 BY 2
4550  023550  020227  172140        4$:     CMP     R2,#172140              ;ALL REGISTERS TESTED?
4551  023554  103772                        BLO     2$                      ;NOPE, TEST SOMEMORE
4552  023556  000426                        BR      98$                     ;TEST IS OVER.
4553                                ;;****  TRAP HANDLER                    ***
4554  023560  012705  001000        66$:    MOV     #1000,R5                ;SET COUNTER AND...
4555  023564  077501                        SOB     R5,.                    ;WAIT A WHILE FOR THE GATES TO SETTLE
4556  023566  005737  177744                TST     @#MSER                  ;ANY BITS SET IN MEM. ERR. REG?
4557  023572  001004                        BNE     68$                     ;YES, ERROR
4558  023574  032737  000020  177766        BIT     #BIT4,@#CPUERR          ;UNIBUS ERROR BIT SET?
4559  023602  001011                        BNE     77$                     ;YES,OK
4560  023604  010237  001160        68$:    MOV     R2, $REG2               ;PRINT OFFENDING CSR
4561  023610  013737  177744  001172        MOV     @#MSER,$TMP1            ;       MSER
4562  023616  013737  177766  001174        MOV     @#CPUERR,$TMP2          ;       CPUERR
4563  023624  104055                        ERROR   55                      ;REPORT ERROR
4564  023626  005037  177766        77$:    CLR     @#CPUERR                ;CLEAR THE ERROR REGISTER
4565  023632  000002                        RTI
4566                                ;;**********************************
4567
4568  023634  012637  000004        98$:    MOV     (SP)+, @#4              ;RESTORE CPU ERROR VECTOR
4569
4570  023640                        99$:
4571                                ;;*************************************************************
4572                                ;*TEST 47        CHECK CACHE BYPASS ON UNIBUS MAP PAGE
4573                                        ;THIS TEST CHECKS CACHE BYPASS ON UNIBUS-MAP-PAGE
4574                                        ;THIS TEST IS EXECUTED ONLY ON KB11-EM OR 11/74          (KB11CM)
4575                                        ;THIS TEST USES MEMORY MANAGMENT
4576                                ;;*************************************************************
4577  023640  000004                TST47:  SCOPE
4578  023642  012737  000005  001204        MOV     #5,$TIMES               ;;DO 5 ITERATIONS
4579
4580  023650  013746  177776                MOV     @#PS,-(SP)
4581  023654  042716  000020                BIC     #20,(SP)                ;CLEAR T BIT IF SET
4582  023660  012746  023666                MOV     #1$,-(SP)
4583  023664  000002                        RTI
4584  023666                        1$:
4585
4586  023666  012737  024360  001324        MOV     #$EOP,NXTTST
4587  023674  105737  001331                TSTB    KB11EM                  ;IS THIS A KB11-EM?
4588  023700  001005                        BNE     CBMP                    ;BR IF YES
4589  023702  105737  001332                TSTB    KB11CM
4590  023706  001002                        BNE     CBMP
4591  023710  000137  024360                JMP     $EOP
4592  023714  012737  170000  172354  CBMP: MOV     #170000,@#KIPAR6                ;MAP TO UNIBUS MAPPING BOX
4593  023722  012737  010406  172314        MOV     #10406,@#KIPDR6                 ;THROUGH PAGE 6
4594
4595  023730  012705  170200                MOV     #MAPL00,R5                      ;STARTING ADDRESS OF UBMR'S
4596
```

```
4597  023734  005037  17,746          CBMPA:  CLR     @#CONTRL
4598  023740  032737  010000  177746    1$:   BIT     #VCIP,@#CONTRL
4599  023746  001374                          BNE     1$
4600
4601  023750  012700  023714                  MOV     #CBMP,R0                ;MAKE TEST CODE HIT IN GROUP 0
4602  023754  012702  001000                  MOV     #1000,R2
4603  023760  012737  000054  177746    2$:   MOV     #S1MOM1,@#CONTRL
4604  023766  005760  002000                  TST     2000(R0)
4605  023772  012737  000034  177746          MOV     #S0MOM1,@#CONTRL
4606  024000  005720                          TST     (R0)+
4607  024002  077212                          SOB     R2,2$           ;DONE?
4608
4609  024004  012700  040060                  MOV     #TSTDAT,R0              ;MAKE TEST DATA HIT IN GROUP 1
4610  024010  012702  001000                  MOV     #1000,R2
4611  024014  012737  000040  177746          MOV     #S1,@#CONTRL
4612  024022  005720                    9$:   TST     (R0)+
4613  024024  077202                          SOB     R2,9$
4614
4615  024026  012737  177600  172356          MOV     #177600,@#KIPAR7        ;MAP I/O PAGE THROUGH PAGE 7
4616  024034  012737  077406  172316          MOV     #77406,@#KIPDR7
4617  024042  005037  172340                  CLR     @#KIPAR0                ;MAP 0-4 K VIRTUAL INTO
4618  024046  012737  077406  172300          MOV     #77406,@#KIPDR0         ;0-4 K PHYSICAL ADDRESS FORCE
4619  024054  012737  000200  172342          MOV     #200,@#KIPAR1           ;MAP 4-8K VIRTUAL INTO 4-8K
4620  024062  012737  077406  172302          MOV     #77406,@#KIPDR1         ;PHYSICAL ADDRESS SPACE
4621  024070  012737  000400  172344          MOV     #400,@#KIPAR2
4622  024076  012737  077406  172304          MOV     #77406,@#KIPDR2
4623
4624  024104  012700  040060                  MOV     #TSTDAT,R0              ;SET UP UNIBUS MAPPING
4625  024110  042700  000077                  BIC     #77,R0                  ;REGISTER. NOTE THIS REGISTER
4626  024114  010025                          MOV     R0,(R5)+                ;WILL BE USED FOR CACHE-
4627  024116  005015                          CLR     (R5)                    ;BYPASS ON UNIBUS MAP PAGE.
4628
4629  024120  012700  040060                  MOV     #TSTDAT,R0              ;FORM THE VIRTUAL ADDRESS FOR
4630  024124  042700  177700                  BIC     #177700,R0              ;THAT DATA.  NOTE TEST DATA
4631  024130  052700  140000                  BIS     #140000,R0              ;IS MAPPED THROUGH PAR6
4632  024134  010001                          MOV     R0,R1                   ;R0 CONTAINS THE VA
4633  024136  012702  001000                  MOV     #1000,R2
4634
4635  024142  012737  000060  172516          MOV     #60,@#MMR3              ;ENABLE 22-BIT MAPPING
4636  024150  012737  000001  177572          MOV     #1,@#MMR0               ;TURN ON KT AND UNIBUS MAP
4637
4638  024156  005710                    3$:   TST     (R0)                    ;REFERENCE TEST-DATA,CHECK IT
4639  024160  032737  000010  177752          BIT     #10,@#HITMIS            ;IS A HIT
4640  024166  001011                          BNE     4$
4641  024170  013737  177746  001154          MOV     @#CONTRL,$REG0
4642  024176  012737  000001  001156          MOV     #1,$REG1                ;GROUP NO. BEING TESTED
4643  024204  010037  001160                  MOV     R0,$REG2                ;TEST-DATA ADDRESS
4644  024210  104056                          ERROR   56                      ;TEST DATA REFERENCE NOT
4645                                                                          ;A HIT, AS EXPECTED. IT WAS
4646                                                                          ;MADE BIT IN GROUP 1.
4647                                                                          ;PREVIOUSLY. THEN IT WAS
4648                                                                          ;REFERENCE THROUGH THE UNIBUS
4649                                                                          ;MAP
4650
4651  024212  062700  000002              4$:   ADD     #2,R0
4652  024216  077221                          SOB     R2,3$                   ;DONE?
```

```
4653
4654  024220  052715  100000                    BIS     #BYP,(R5)       ;GET BYPASS IN (HI) UNIBUS
4655                                                                      ;MAP REGISTER BEING CHECKED
4656  024224  012702  001000                    MOV     #1000,R2
4657  024230  010100                            MOV     R1,R0           ;R0 CONTAINS THE VA OF TEST-DATA
4658  024232  005711                      5$:   TST     (R1)            ;REFERENCE TEST DATA THROUGH
4659                                                                      ;THE UNIBUS MAP (WITH CACHE
4660                                                                      ;BYPASS SET) AND CHECK
4561                                                                      ;IF THE REFERENCE WAS A MISS
4662
4663  024234  062701  000002                6$:   ADD     #2,R1
4664  024240  077204                            SOB     R2,5$           ;DONE?
4665
4666  024242  005037  177572                    CLR     @#MMR0          ;TURN OFF MEM MNGMNT
4667  024246  005037  172516                    CLR     @#MMR3
4668  024252  042715  100000                    BIC     #BYP,(R5)       ;CLEAR CACHE BYPASS IN UBMR
4669  024256  012700  040060                    MOV     #TSTDAT,R0
4670  024262  012702  000400                    MOV     #400,R2
4671  024266  005710                      7$:   TST     (R0)            ;REFERENCE TEST-DATA AGAIN
4672                                                                      ;AND CHECK THAT IT WAS
4673                                                                      ;INVALIDATED PREVIOUSLY
4674  024270  032737  000010  177752            BIT     #10,@#HITMIS    ;MISS?
4675  024276  001413                            BEQ     8$              ;YES
4676  024300  013737  177746  001154            MOV     @#CONTRL,$REG0
4677  024306  010537  001156                    MOV     R5,$REG1        ;ADDRESS OF UB MAP REGISTER THAT
4678  024312  162737  000002  001156            SUB     #2,$REG1        ;IS BEING TESTED
4679  024320  010037  001160                    MOV     R0,$REG2        ;TEST-DATA ADDRESS
4680  024324  104057                            ERROR   57              ;TEST DATA REFERENCE WAS NOT
4681                                                                      ;A MISS.  TEST-DATA WAS
4682                                                                      ;PREVIOUSLY REFERENCED IN
4683                                                                      ;BYPASS (ON UBMR) MODE,HENCE
4684                                                                      ;IT SHOULD HAVE BEEN
4685                                                                      ;INVALIDATED.  BUT, SUBSEQUENT
4686                                                                      ;REFERENCE WAS NOT A MISS,
4687                                                                      ;AS EXPECTED
4688
4689  024326  062700  000004                8$:   ADD     #4,R0
4690  024332  077223                            SOB     R2,7$           ;DONE?
4691  024334  020527  170366                    CMP     R5,#MAPH35      ;DONE CHECKING ALL 31 UBMR'S
4692  024340  001407                            BEQ     10$
4693
4694  024342  062705  000002                    ADD     #2,R5           ;ADDRESS OF NEXT UBMR
4695  024346  062737  000200  172354            ADD     #200,@#KIPAR6   ;MAP  PAR 6 TO RESET THE
4696  024354  000137  023734                    JMP     CBMPA           ;NEXT UBMR (TO BE TESTED)
4697  024360                            10$:
4698
4699                                            ;;**************************************************************
4700
4701                                            .SBTTL  END OF PASS ROUTINE
4702
4703                                            ;*INCREMENT THE PASS NUMBER ($PASS)
4704                                            ;*INDICATE END-OF-PROGRAM AFTER 1 PASSES THRU THE PROGRAM
4705                                            ;*TYPE 'END PASS #XXXXX TOTAL NUMBER OF ERRORS SINCE LAST REPORT YYYYY''
4706                                            ;*WHERE XXXXX AND YYYYY ARE DECIMAL NUMBERS
4707                                            ;*IF SW12=1 INHIBIT TRACE TRAP
4708                                            ;*IF THERES A MONITOR GO TO IT
```

```
4709
4710
4711  024360
4712  024360  000004
4713  024362  005037  177572
4714  024366  005037  172516
4715  024372  10442C
4716  024374  005037  001102
4717  024400  005037  001204
4718  024404  005237  001100
4719  024410  042737  100000  001100
4720  024416  005327
4721  024420  000001
4722  024422  003072
4723  024424  012737
4724  024426  000001
4725  024430  024420
4726  024432  104400  024440
4727  024436  000407
4728
4729  024456
4730  024456  013746  001100
4731
4732  024462  104410
4733  024464  104400  024472
4734  024470  000421
4735
4736  024534
4737  024534  013746  001112
4738
4739  024540  104410
4740  024542  104400  001215
4741  024546  005037  001112
4742  024552  013700  000042
4743  024556  001414
4744  024560  005046
4745    4562  012746  024570
4746  024566  000427
4747
4748  024570
4749  024570  013700  000042
4750  024574  001405
4751  024576  000005
4752  024600  004710
4753  024602  000240
4754  024604  000240
4755  024606  000240
4756  024610
4757  024610  013746  177776
4758  024614  042716  000020
4759  024620  032737  010000  177570
4760  024626  001005
4761  024630  005137  024654
4762  024634  100402
4763  024636  052716  000020
4764  024642  012746  024650

;*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 $ERTTL FOR TYPEOUT
                                        ;;TOTAL NUMBER OF ERRORS
        TYPDS                           ;;GO TYPE--DECIMAL ASCII WITH SIGN
        TYPE    ,$CRLF                  ;;TYPE CARRIAGE RETURN, LINE FEED
        CLR     $ERTTL                  ;;CLEAR ERROR TOTAL
$GET42: MOV     @#42,R0                 ;;GET MONITOR ADDRESS
        BEQ     $DOAGN                  ;;BRANCH IF NO MONITOR
        CLR     -(SP)                   ;;INSURE THE "T" BIT IS CLEAR
        MOV     #$CLR.T,-(SP)           ;;SETUP FOR AN RTI OR RTT
        BR      $RTRN                   ;;GO DO AN RTI OR RTT TO LOAD THE PSW
                                        ;;WITH A CLEARED "T" BIT
$CLR.T:
        MOV     @#42,R0                 ;;INSURE R0 CONTAINS THE MONITORS
        BEQ     $DOAGN                  ;;RETURN ADDRESS
        RESET                           ;;CLEAR THE WORLD
$ENDAD: JSR     PC,(R0)                 ;;GO TO MONITOR
        NOP                             ;;SAVE ROOM
        NOP                             ;;FOR
        NOP                             ;;ACT11
$DOAGN:
        MOV     @#PS,-(SP)              ;;PUT THE PS ON THE STACK AND
        BIC     #20,(SP)                ;;CLEAR THE "T" BIT
        BIT     #BIT12,@#SWR            ;;RUN WITH TRACE TRAP?
        BNE     1$                      ;;BR IF NO
        COM     $TBIT                   ;;IS IT TIME FOR TRACE TRAP
        BMI     1$                      ;;BR IF NO
        BIS     #20,(SP)                ;;SET TRACE TRAP
1$:     MOV     #$LOOP,-(SP)            ;;JUMP TO START OF TEST
```

```
4765  024646  000002                  $RTRN:  RTI                             ;;RETURN--THIS IS CHANGED TO
4766                                                                          ;;AN 'RTT' IF 'RTT' IS A LEGAL
4767                                                                          ;;INSTRUCTION
4768  024650                          $LOOP:
4769  024650  000137  010652                  JMP     @#LOOP                 ;;RETURN
4770  024654  000000                  $TBIT:  .WORD   0                      ;;'T' BIT STATE INDICATOR
4771  024656     377     377     000  $ENULL: .BYTE   -1,-1,0                ;;NULL CHARACTER STRING
4772           024662                          .EVEN
4773
4774                                  ;MESSAGES
4775  024662  041600  052520  052440  MSG1:   .ASCIZ<CRLF>    ''CPU UNDER TEST FOUND TO BE A ''
4776  024670  042116  051105  052040
4777  024676  051505  020124  047506
4778  024704  047125  020104  047524
4779  024712  041040  020105  020101
4780  024720     000
4781  024721     113  030502  026461  MSG2:   .ASCIZ  'KB11-EM'<CRLF>
4782  024726  046505  000200
4783  024732  041113  030461  041055  MSG3:   .ASCIZ  'KB11-B/C'<CRLF>
4784  024740  041457  000200
4785  024744  041113  030461  041455  MSG4:   .ASCIZ  'KB11-CM                ''<CRLF>
4786  024752  020115  020040  020040
4787  024760  020040  020040  020040
4788  024766  020040  020040  100040
4789  024774     000
4790  024775     113  030502  026461  MSG5:   .ASCIZ  'KB11-E'<CRLF>
4791  025002  100105     000
4792
4793
4794                                          .SBTTL  ERROR MESSAGES AND DATA TABLES
4795  025005     116  052117  052040  EM1:    .ASCIZ  ?NOT THE CORRECT TRAP CONDITION THRU ERRVEC (#004)?
4796  025012  042510  041440  051117
4797  025020  042522  052103  052040
4798  025026  040522  020120  047503
4799  025034  042116  052111  047511
4800  025042  020116  044124  052522
4801  025050  042440  051122  042526
4802  025056  020103  021450  030060
4803  025064  024464     000
4804  025067     125  042516  050130  EM2:    .ASCIZ  ?UNEXPECTED CPU TRAP THRU ERRVEC (#004)?
4805  025074  041505  042524  020104
4806  025102  050103  020125  051124
4807  025110  050101  052040  051110
4808  025116  020125  051105  053122
4809  025124  041505  024040  030043
4810  025132  032060  000051
4811  025136  047125  054105  042520  EM3:    .ASCIZ  ?UNEXPECTED CACHE PARITY ERROR THRU CACHVEC, WILL RETRY TEST ONCE?
4812  025144  052103  042105  041440
4813  025152  041501  042510  050040
4814  025160  051101  052111  020131
4815  025166  051105  047522  020122
4816  025174  044124  052522  041440
4817  025202  041501  053110  041505
4818  025210  020054  044527  046114
4819  025216  051040  052105  054522
4820  025224  052040  051505  020124
```

```
4821  025232  047117  042503  000
4822  025237  125     042516  050130  EM4:    .ASCIZ  ?UNEXPECTED MAIN MEMORY PARITY ERROR THRU CACHVEC, WILL RETRY TEST ONCE?
4823  025244  041505  042524  020104
4824  025252  040515  047111  046440
4825  025260  046505  051117  020131
4826  025266  040520  044522  054524
4827  025274  042440  051122  051117
4828  025302  052040  051110  020125
4829  025310  040503  044103  042526
4830  025316  026103  053440  046111
4831  025324  020114  042522  051124
4832  025332  020131  042524  052123
4833  025340  047440  041516  000105
4834  025346  047125  054105  042520  EM5:    .ASCIZ  ?UNEXPECTED MEMORY MANAGEMENT TRAP, MEMORY MANAGEMENT STATUS REGISTERS?
4835  025354  052103  042105  046440
4836  025362  046505  051117  020131
4837  025370  040515  040516  042507
4838  025376  042515  052116  052040
4839  025404  040522  026120  046440
4840  025412  046505  051117  020131
4841  025420  040515  040516  042507
4842  025426  042515  052116  051440
4843  025434  040524  052524  020123
4844  025442  042522  044507  052123
4845  025450  051105  000123
4846  025454  052523  046515  051101  EM6:    .ASCIZ  ?SUMMARY OF MAP REGISTERS THAT TIMED OUT ON READ?
4847  025462  020131  043117  046440
4848  025470  050101  051040  043505
4849  025476  051511  042524  051522
4850  025504  052040  040510  020124
4851  025512  044524  042515  020104
4852  025520  052517  020124  047117
4853  025526  051040  040505  000104
4854  025534  052523  046515  051101  EM7:    .ASCIZ  ?SUMMARY OF CACHE REGISTERS THAT TIMED OUT ON READ?
4855  025542  020131  043117  041440
4856  025550  041501  042510  051040
4857  025556  043505  051511  042524
4858  025564  051522  052040  040510
4859  025572  020124  044524  042515
4860  025600  020104  052517  020124
4861  025606  047117  051040  040505
4862  025614  000104
4863  025616  052523  046515  051101  EM10:   .ASCIZ  ?SUMMARY OF MAP REGISTERS NOT HOLDING ZERO IN LOW 16 BITS?
4864  025624  020131  043117  046440
4865  025632  050101  051040  043505
4866  025640  051511  042524  051522
4867  025646  047040  052117  044040
4868  025654  046117  044504  043516
4869  025662  055040  051105  020117
4870  025670  047111  046040  053517
4871  025676  030440  020066  044502
4872  025704  051524  000
4873  025707  123     046525  040515  EM11:   .ASCIZ  ?SUMMARY OF MAP REGISTERS NOT HOLDING ZERO IN UPPER 6 BITS?
4874  025714  054522  047440  020106
4875  025722  040515  020120  042522
4876  025730  044507  052123  051105
```

```
4877  025736  020123  047516  020124
4878  025744  047510  042114  047111
4879  025752  020107  042532  047522
4880  025760  044440  020116  050125
4881  025766  042520  020122  020066
4882  025774  044502  051524     000
4883  026001     120  051517  044523  EM12:   .ASCIZ  ?POSSIBLE ERROR IN MAP REGISTER DATA PATH (MAP REG 00)?
4884  026006  046102  020105  051105
4885  026014  047522  020122  047111
4886  026022  046440  050101  051040
4887  026030  043505  051511  042524
4888  026036  020122  040504  040524
4889  026044  050040  052101  020110
4890  026052  046450  050101  051040
4891  026060  043505  030040  024460
4892  026066     000
4893  026067     116  053517  050040  EM13:   .ASCIZ  ?NOW PROBABLE ERROR IN MAP REGISTER DATA PATH (MAP REG 20)?
4894  026074  047522  040502  046102
4895  026102  020105  051105  047522
4896  026110  020122  047111  046440
4897  026116  050101  051040  043505
4898  026124  051511  042524  020122
4899  026132  040504  040524  050040
4900  026140  052101  020110  046450
4901  026146  050101  051040  043505
4902  026154  031040  024460     000
4903  026161     123  046525  040515  EM14:   .ASCIZ  ?SUMMARY OF DUAL ADDRESSING ERRORS ON LOADING MAP REGISTERS?
4904  026166  054522  047440  020106
4905  026174  052504  046101  040440
4906  026202  042104  042522  051523
4907  026210  047111  020107  051105
4908  026216  047522  051522  047440
4909  026224  020116  047514  042101
4910  026232  047111  020107  040515
4911  026240  020120  042522  044507
4912  026246  052123  051105  000123
4913  026254  052523  046515  051101  EM15:   .ASCIZ  ?SUMMARY OF COUNT PATTERN FAILURES IN LOWER 16 BITS OF MAP REGISTERS?
4914  026262  020131  043117  041440
4915  026270  052517  052116  050040
4916  026276  052101  042524  047122
4917  026304  043040  044501  052514
4918  026312  042522  020123  047111
4919  026320  046040  053517  051105
4920  026326  030440  020066  044502
4921  026334  051524  047440  020106
4922  026342  040515  020120  042522
4923  026350  044507  052123  051105
4924  026356  000123
4925  026360  052523  046515  051101  EM16:   .ASCIZ  ?SUMMARY OF COUNT PATTERN FAILURES IN UPPER 6 BITS OF MAP REGISTERS?
4926  026366  020131  043117  041440
4927  026374  052517  052116  050040
4928  026402  052101  042524  047122
4929  026410  043040  044501  052514
4930  026416  042522  020123  047111
4931  026424  052440  050120  051105
4932  026432  033040  041040  052111
```

```
 4933  026440  020123  043117  046440
 4934  026446  050101  051040  043505
 4935  026454  051511  042524  051522
 4936  026462     000
 4937  026463     103  052517  042114  EM17:   .ASCII  ?COULD NOT CLEAR CACHE CONTROL REGISTER?<CRLF>
 4938  026470  047040  052117  041440
 4939  026476  042514  051101  041440
 4940  026504  041501  042510  041440
 4941  026512  047117  051124  046117
 4942  026520  051040  043505  051511
 4943  026526  042524  100122
 4944  026532  047520  051523  041111          .ASCIZ  ?POSSIBLE ERROR IN CACHE REGISTER DATA PATH?
 4945  026540  042514  042440  051122
 4946  026546  051117  044440  020116
 4947  026554  040503  044103  020105
 4948  026562  042522  044507  052123
 4949  026570  051105  042040  052101
 4950  026576  020101  040520  044124
 4951  026604     000
 4952  026605     103  052517  042114  EM20:   .ASCII  ?COULD NOT CLEAR CACHE MAINTENENCE REGISTER?<CRLF>
 4953  026612  047040  052117  041440
 4954  026620  042514  051101  041440
 4955  026626  041501  042510  046440
 4956  026634  044501  052116  047105
 4957  026642  047105  042503  051040
 4958  026650  043505  051511  042524
 4959  026656  100122
 4960  026660  047520  051523  041111          .ASCIZ  ?POSSIBLE ERROR IN CACHE REGISTER DATA PATH?
 4961  026666  042514  042440  051122
 4962  026674  051117  044440  020116
 4963  026702  040503  044103  020105
 4964  026710  042522  044507  052123
 4965  026716  051105  042040  052101
 4966  026724  020101  040520  044124
 4967  026732     000
 4968  026733     103  052517  042114  EM21:   .ASCII  ?COULD NOT READ 177740 FROM CACHE LO ADDRESS REG (LOADRS)?<CRLF>
 4969  026740  047040  052117  051040
 4970  026746  040505  020104  033461
 4971  026754  033467  030064  043040
 4972  026762  047522  020115  040503
 4973  026770  044103  020105  047514
 4974  026776  040440  042104  042522
 4975  027004  051523  051040  043505
 4976  027012  024040  047514  042101
 4977  027020  051522  100051
 4978  027024  047520  051523  041111          .ASCIZ  ?POSSIBLE ERROR IN CACHE REGISTER DATA PATH?
 4979  027032  042514  042440  051122
 4980  027040  051117  044440  020116
 4981  027046  040503  044103  020105
 4982  027054  042522  044507  052123
 4983  027062  051105  042040  052101
 4984  027070  020101  040520  044124
 4985  027076     000
 4986  027077     103  052517  042114  EM22:   .ASCII  ?COULD NOT READ 000003 FROM CACHE HI ADDRESS REG (HIADRS)?<CRLF>
 4987  027104  047040  052117  051040
 4988  027112  040505  020104  030060
```

```
4989  027120  030060  031460  043040
4990  027126  047522  020115  040503
4991  027134  044103  020105  044510
4992  027142  040440  042104  042522
4993  027150  051523  051040  043505
4994  027156  024040  044510  042101
4995  027164  051522  100051
4996  027170  047520  051523  041111          .ASCIZ  ?POSSIBLE ERROR IN CACHE REGISTER DATA PATH?
4997  027176  042514  042440  051122
4998  027204  051117  044440  020116
4999  027212  040503  044103  020105
5000  027220  042522  044507  052123
5001  027226  051105  042040  052101
5002  027234  020101  040520  044124
5003  027242     000
5004  027243     123  046525  040515  EM23:   .ASCIZ  ?SUMMARY OF COUNT PATTERN FAILURES IN CACHE CONTROL REGISTER?
5005  027250  054522  047440  020106
5006  027256  047503  047125  020124
5007  027264  040520  052124  051105
5008  027272  020116  040506  046111
5009  027300  051125  051505  044440
5010  027306  020116  040503  044103
5011  027314  020105  047503  052116
5012  027322  047522  020114  042522
5013  027330  044507  052123  051105
5014  027336     000
5015  027337     123  046525  040515  EM24:   .ASCIZ  ?SUMMARY OF COUNT PATTERN FAILURES IN CACHE MAINTENENCE REGISTER?
5016  027344  054522  047440  020106
5017  027352  047503  047125  020124
5018  027360  040520  052124  051105
5019  027366  020116  040506  046111
5020  027374  051125  051505  044440
5021  027402  020116  040503  044103
5022  027410  020105  040515  047111
5023  027416  042524  042516  041516
5024  027424  020105  042522  044507
5025  027432  052123  051105     000
5026  027437     122  043105  051105  EM25:   .ASCIZ  ?REFERENCED MAP REGISTER 0 WITH ADDRESS ONE BIT DIFFERENT THAN 770200?
5027  027444  047105  042503  020104
5028  027452  040515  020120  042522
5029  027460  044507  052123  051105
5030  027466  030040  053440  052111
5031  027474  020110  042101  051104
5032  027502  051505  020123  047117
5033  027510  020105  044502  020124
5034  027516  044504  043106  051105
5035  027524  047105  020124  044124
5036  027532  047101  033440  030067
5037  027540  030062  000060
5038  027544  042522  042506  042522  EM26:   .ASCII  ?REFERENCED CACHE LOW ADDRESS REGISTER WITH ADDRESS ONE BIT?<CRLF>
5039  027552  041516  042105  041[illegible]0
5040  027560  041501  042510  046040
5041  027566  053517  040440  042104
5042  027574  042522  051523  051040
5043  027602  043505  051511  042524
5044  027610  020122  044527  044124
```

```
5045  027616  040440  042104  042522
5046  027624  051523  047440  042516
5047  027632  041040  052111     200
5048  027637     104  043111  042506          .ASCIZ  ?DIFFERENT THAN 777740?
5049  027644  042522  052116  052040
5050  027652  040510  020116  033467
5051  027660  033467  030064     000
5052  027665     103  047101  052047  EM30:   .ASCII  ?CAN'T GET TO MAIN MEMORY FROM UNIBUS WITH THE MAP OFF?<CRLF>
5053  027672  043440  052105  052040
5054  027700  020117  040515  047111
5055  027706  046440  046505  051117
5056  027714  020131  051106  046517
5057  027722  052440  044516  052502
5058  027730  020123  044527  044124
5059  027736  052040  042510  046440
5060  027744  050101  047440  043106
5061  027752     200
5062  027753     123  020117  023511          .ASCIZ  ?SO I'LL JUMP TO THE SIZE JUMPER TEST FOR VERIFICATION?
5063  027760  046114  045040  046525
5064  027766  020120  047524  052040
5065  027774  042510  051440  055111
5066  030002  020105  052512  050115
5067  030010  051105  052040  051505
5068  030016  020124  047506  020122
5069  030024  042526  044522  044506
5070  030032  040503  044524  047117
5071  030040     000
5072  030041     123  046525  040515  EM31:   .ASCIZ  ?SUMMARY OF COUNT PATTERN FAILURES ON THE UNIBUS DATA PATH?
5073  030046  054522  047440  020106
5074  030054  047503  047125  020124
5075  030062  040520  052124  051105
5076  030070  020116  040506  046111
5077  030076  051125  051505  047440
5078  030104  020116  044124  020105
5079  030112  047125  041111  051525
5080  030120  042040  052101  020101
5081  030126  040520  044124     000
5082  030133     125  044516  052502  EM32:   .ASCIZ  ?UNIBUS MAP IS RELOCATING WHEN NOT ENABLED?
5083  030140  020123  040515  020120
5084  030146  051511  051040  046105
5085  030154  041517  052101  047111
5086  030162  020107  044127  047105
5087  030170  047040  052117  042440
5088  030176  040516  046102  042105
5089  030204     000
5090  030205     103  047101  047516  EM33:   .ASCII  ?CANNOT USE ANY OF THE MAP REGISTERS, OR PHYSICAL?<CRLF>
5091  030212  020124  051525  020105
5092  030220  047101  020131  043117
5093  030226  052040  042510  046440
5094  030234  050101  051040  043505
5095  030242  051511  042524  051522
5096  030250  020054  051117  050040
5097  030256  054510  044523  040503
5098  030264  100114
5099  030266  042101  051104  051505          .ASCIZ  ?ADDRESS BIT14 IS STUCK LOW. MUST RESTART PROGRAM IF NO LOOP?
5100  030274  020123  044502  030524
```

```
5101  030302  020064  051511  051440
5102  030310  052524  045503  046040
5103  030316  053517  020056  052515
5104  030324  052123  051040  051505
5105  030332  040524  052122  050040
5106  030340  047522  051107  046501
5107  030346  044440  020106  047516
5108  030354  046040  047517  000120
5109  030362  044124  020105  052516  EM34:   .ASCII  ?THE NUMBER OF MAP REGISTERS REMOVED BY JUMPER SETTING DOES?<CRLF>
5110  030370  041115  051105  047440
5111  030376  020106  040515  020120
5112  030404  042522  044507  052123
5113  030412  051105  020123  042522
5114  030420  047515  042526  020104
5115  030426  054502  045040  046525
5116  030434  042520  020122  042523
5117  030442  052124  047111  020107
5118  030450  047504  051505     200
5119  030455     116  052117  040440        .ASCIZ  ?NOT AGREE WITH THE NUMBER FOUND TO BE MISSING?
5120  030462  051107  042505  053440
5121  030470  052111  020110  044124
5122  030476  020105  052516  041115
5123  030504  051105  043040  052517
5124  030512  042116  052040  020117
5125  030520  042502  046440  051511
5126  030526  044523  043516     000
5127  030533     124  042510  051440  EM35:   .ASCII  ?THE SIZE JUMPERS ON THE UNIBUS MAP ARE NOT SET IN?<CRLF>
5128  030540  055111  020105  052512
5129  030546  050115  051105  020123
5130  030554  047117  052040  042510
5131  030562  052440  044516  052502
5132  030570  020123  040515  020120
5133  030576  051101  020105  047516
5134  030604  020124  042523  020124
5135  030612  047111     200
5136  030615     124  042510  051111        .ASCII  ?THEIR DEFAULT POSITION, WHICH ALLOWS UNIBUS ADDRESSES?<CRLF>
5137  030622  042040  043105  052501
5138  030630  052114  050040  051517
5139  030636  052111  047511  026116
5140  030644  053440  044510  044103
5141  030652  040440  046114  053517
5142  030660  020123  047125  041111
5143  030666  051525  040440  042104
5144  030674  042522  051523  051505
5145  030702     200
5146  030703     060  030060  030060        .ASCII  ?000000 TO 757776 TO REFERENCE MAIN MEMORY?<CRLF>
5147  030710  020060  047524  033440
5148  030716  033465  033467  020066
5149  030724  047524  051040  043105
5150  030732  051105  047105  042503
5151  030740  046440  044501  020116
5152  030746  042515  047515  054522
5153  030754     200
5154  030755     124  042510  051111        .ASCIZ  ?THEIR CURRENT SETTING ALLOWS ONLY:?
5155  030762  041440  051125  042522
5156  030770  052116  051440  052105
```

```
5157  030776  044524  043516  040440
5158  031004  046114  053517  020123
5159  031012  047117  054514  000072
5160  031020  040515  020120  042522  EM36:   .ASCIZ  ?MAP REGISTER UNDER TEST DID NOT RESPOND IN DUAL MAPPING TEST?
5161  031026  044507  052123  051105
5162  031034  052440  042116  051105
5163  031042  052040  051505  020124
5164  031050  044504  020104  047516
5165  031056  020124  042522  050123
5166  031064  047117  020104  047111
5167  031072  042040  040525  020114
5168  031100  040515  050120  047111
5169  031106  020107  042524  052123
5170  031114     000
5171  031115     123  046525  040515  EM37:   .ASCIZ  ?SUMMARY OF UNIBUS ADDRESS ERRORS, WITH THE MAP RELOCATION DISABLED?
5172  031122  054522  047440  020106
5173  031130  047125  041111  051525
5174  031136  040440  042104  042522
5175  031144  051523  042440  051122
5176  031152  051117  026123  053440
5177  031160  052111  020110  044124
5178  031166  020105  040515  020120
5179  031174  042522  047514  040503
5180  031202  044524  047117  042040
5181  031210  051511  041101  042514
5182  031216  000104
5183  031220  040515  047111  046440  EM40:   .ASCIZ  ?MAIN MEMORY TIMEOUT OVER THE UNIBUS DID NOT OCCUR PROPERLY?
5184  031226  046505  051117  020131
5185  031234  044524  042515  052517
5186  031242  020124  053117  051105
5187  031250  052040  042510  052440
5188  031256  044516  052502  020123
5189  031264  044504  020104  047516
5190  031272  020124  041517  052503
5191  031300  020122  051120  050117
5192  031306  051105  054514     000
5193  031313     122  046105  041517  EM41:   .ASCIZ  ?RELOCATION THRU THE MAP WAS NOT CORRECT, FULL ADD?
5194  031320  052101  047511  020116
5195  031326  044124  052522  052040
5196  031334  042510  046440  050101
5197  031342  053440  051501  047040
5198  031350  052117  041440  051117
5199  031356  042522  052103  020054
5200  031364  052506  046114  040440
5201  031372  042104     000
5202  031375     122  046105  041517  EM42:   .ASCIZ  ?RELOCATION THRU THE MAP WAS NOT CORRECT, CARRY PROPAGATION?
5203  031402  052101  047511  020116
5204  031410  044124  052522  052040
5205  031416  042510  046440  050101
5206  031424  053440  051501  047040
5207  031432  052117  041440  051117
5208  031440  042522  052103  020054
5209  031446  040503  051122  020131
5210  031454  051120  050117  043501
5211  031462  052101  047511  000116
5212  031470  044124  020105  047524  EM43:   .ASCIZ  ?THE TOP OF MEMORY IS DIFFERENT THAN THE SIZE JUMPERS?
```

```
5213  031476  020120  043117  046440
5214  031504  046505  051117  020131
5215  031512  051511  042040  043101
5216  031520  042506  042522  052116
5217  031526  052040  040510  020116
5218  031534  044124  020105  044523
5219  031542  042532  045040  046525
5220  031550  042520  051522     000
5221  031555     120  051101  052111  EM44:   .ASCIZ  ?PARITY REPORTING THRU THE MAP IS NOT CORRECT?
5222  031562  020131  042522  047520
5223  031570  052122  047111  020107
5224  031576  044124  052522  052040
5225  031604  042510  046440  050101
5226  031612  044440  020123  047516
5227  031620  020124  047503  051122
5228  031626  041505  000124
5229  031632  040515  047111  046440  EM45:   .ASCII  ?MAIN MEMORY TIMEOUT OVER THE UNIBUS DID NOT OCCUR PROPERLY?<CRLF>
5230  031640  046505  051117  020131
5231  031646  044524  042515  052517
5232  031654  020124  053117  051105
5233  031662  052040  042510  052440
5234  031670  044516  052502  020123
5235  031676  044504  020104  047516
5236  031704  020124  041517  052503
5237  031712  020122  051120  050117
5238  031720  051105  054514     200
5239  031725     124  051505  020124          .ASCIZ  ?TEST CODE BEING RUN OVER UNIBUS?
5240  031732  047503  042504  041040
5241  031740  044505  043516  051040
5242  031746  047125  047440  042526
5243  031754  020122  047125  041111
5244  031762  051525     000
5245  031765     122  046105  041517  EM46:   .ASCII  ?RELOCATION THRU THE MAP WAS NOT CORRECT, FULL ADD?<CRLF>
5246  031772  052101  047511  020116
5247  032000  044124  052522  052040
5248  032006  042510  046440  050101
5249  032014  053440  051501  047040
5250  032022  052117  041440  051117
5251  032030  042522  052103  020054
5252  032036  052506  046114  040440
5253  032044  042104     200
5254  032047     124  051505  020124          .ASCIZ  ?TEST CODE BEING RUN OVER UNIBUS?
5255  032054  047503  042504  041040
5256  032062  044505  043516  051040
5257  032070  047125  047440  042526
5258  032076  020122  047125  041111
5259  032104  051525     000
5260  032107     122  046105  041517  EM47:   .ASCII  ?RELOCATION THRU THE MAP WAS NOT CORRECT, CARRY PROPAGATION?<CRLF>
5261  032114  052101  047511  020116
5262  032122  044124  052522  052040
5263  032130  042510  046440  050101
5264  032136  053440  051501  047040
5265  032144  052117  041440  051117
5266  032152  042522  052103  020054
5267  032160  040503  051122  020131
5268  032166  051120  050117  043501
```

```
5269  032174  052101  047511  100116
5270  032202  042524  052123  041440          .ASCIZ  ?TEST CODE BEING RUN OVER UNIBUS?
5271  032210  042117  020105  042502
5272  032216  047111  020107  052522
5273  032224  020116  053117  051105
5274  032232  052440  044516  052502
5275  032240  000123
5276  032242  044124  020105  047524  EM50:   .ASCII  ?THE TOP OF MEMORY IS DIFFERENT THAN THE SIZE JUMPERS?<CRLF>
5277  032250  020120  043117  046440
5278  032256  046505  051117  020131
5279  032264  051511  042040  043111
5280  032272  042506  042522  052116
5281  032300  052040  040510  020116
5282  032306  044124  020105  044523
5283  032314  042532  045040  046525
5284  032322  042520  051522     200
5285  032327     124  051505  020124          .ASCIZ  ?TEST CODE BEING RUN OVER UNIBUS?
5286  032334  047503  042504  041040
5287  032342  044505  043516  051040
5288  032350  047125  047440  042526
5289  032356  020122  047125  041111
5290  032364  051525     000
5291  032367     120  051101  052111  EM51:   .ASCII  ?PARITY REPORTING THRU THE MAP IS NOT CORRECT?<CRLF>
5292  032374  020131  042522  047520
5293  032402  052122  047111  020107
5294  032410  044124  052522  052040
5295  032416  042510  046440  050101
5296  032424  044440  020123  047516
5297  032432  020124  047503  051122
5298  032440  041505  100124
5299  032444  042524  052123  041440          .ASCIZ  ?TEST CODE BEING RUN OVER UNIBUS?
5300  032452  042117  020105  042502
5301  032460  047111  020107  052522
5302  032466  020116  053117  051105
5303  032474  052440  044516  052502
5304  032502  000123
5305  032504  052523  046515  051101  EM52:   .ASCIZ  ?SUMMARY OF DUAL MAPPING ERRORS?
5306  032512  020131  043117  042040
5307  032520  040525  020114  040515
5308  032526  050120  047111  020107
5309  032534  051105  047522  051522
5310  032542     000
5311  032543     102  050131  041040  EM53:   .ASCIZ  /BYP BIT IN UBMR COULD NOT BE CLEARED/
5312  032550  052111  044440  020116
5313  032556  041125  051115  041440
5314  032564  052517  042114  047040
5315  032572  052117  041040  020105
5316  032600  046103  040505  042522
5317  032606  000104
5318  032610  054502  020120  044502  EM54:   .ASCIZ  /BYP BIT IN UBMR COULD NOT BE SET/
5319  032616  020124  047111  052440
5320  032624  046502  020122  047503
5321  032632  046125  020104  047516
5322  032640  020124  042502  051440
5323  032646  052105     000
5324  032651     124  040522  020120  EM55:   .ASCIZ  ?TRAP ON MX11 CSR ACCESS CAUSED UNIBUS ERROR?
```

```
 5325  032656  047117  046440  030513
 5326  032664  020061  051503  020122
 5327  032672  041501  042503  051523
 5328  032700  041440  052501  042523
 5329  032706  020104  047125  041111
 5330  032714  051525  042440  051122
 5331  032722  051117     000
 5332  032725     124  051505  020124  EM56:   .ASCIZ  ?TEST DATA REFERENCE NOT A HIT?
 5333  032732  040504  040524  051040
 5334  032740  043105  051105  047105
 5335  032746  042503  047040  052117
 5336  032754  040440  044040  052111
 5337  032762     000
 5338  032763     124  051505  020124  EM57:   .ASCII  ?TEST DATA REFERENCE NOT A MISS?<15><12>
 5339  032770  040504  040524  051040
 5340  032776  043105  051105  047105
 5341  033004  042503  047040  052117
 5342  033012  040440  046440  051511
 5343  033020  006523     012
 5344  033023     103  041501  042510          .ASCIZ  ?CACHE BYPASS ON UNIBUS MAP PAGE DID NOT INVALIDATE CACHED DATA?
 5345  033030  041040  050131  051501
 5346  033036  020123  047117  052440
 5347  033044  044516  052502  020123
 5348  033052  040515  020120  040520
 5349  033060  042507  042040  042111
 5350  033066  047040  052117  044440
 5351  033074  053116  046101  042111
 5352  033102  052101  020105  040503
 5353  033110  044103  042105  042040
 5354  033116  052101  000101
 5355  033122  044124  020105  047506  EM201:  .ASCIZ  ?THE FOLLOWING REGISTERS TIMED OUT WHEN READ?
 5356  033130  046114  053517  047111
 5357  033136  020107  042522  044507
 5358  033144  052123  051105  020123
 5359  033152  044524  042515  020104
 5360  033160  052517  020124  044127
 5361  033166  047105  051040  040505
 5362  033174  000104
 5363  033176  044124  020105  047506  EM202:  .ASCIZ  ?THE FOLLOWING MAP REGISTERS WILL NOT CLEAR?
 5364  033204  046114  053517  047111
 5365  033212  020107  040515  020120
 5366  033220  042522  044507  052123
 5367  033226    1105  020123  044527
 5368  033234   46114  047040  052117
 5369  033242  041440  042514  051101
 5370  033250     000
 5371  033251     124  042510  043040  EM203:  .ASCIZ  ?THE FOLLOWING ARE DUAL ADDRESSING ERRORS IN THE UNIBUS MAP?
 5372  033256  046117  047514  044527
 5373  033264  043516  040440  042522
 5374  033272  042040  040525  020114
 5375  033300  042101  051104  051505
 5376  033306  044523  043516  042440
 5377  033314  051122  051117  020123
 5378  033322  047111  052040  042510
 5379  033330  052440  044516  052502
 5380  033336  020123  040515  000120
```

```
5381  033344  044124  020105  047503  EM204:  .ASCIZ  ?THE COUNT PATTERN THRU THE MAP REGISTERS FAILED?
5382  033352  047125  020124  040520
5383  033360  052124  051105  020116
5384  033366  044124  052522  052040
5385  033374  042510  046440  050101
5386  033402  051040  043505  051511
5387  033410  042524  051522  043040
5388  033416  044501  042514  000104
5389  033424  047125  041111  051525  EM205   .ASCIZ  ?UNIBUS DATA PATH COUNT PATTERN FAILURE?
5390  033432  042040  052101  020101
5391  033440  040520  044124  041440
5392  033446  052517  052116  050040
5393  033454  052101  042524  047122
5394  033462  043040  044501  052514
5395  033470  042522     000
5396  033473     125  044516  052502  EM206:  .ASCIZ  ?UNIBUS ADDRESSING ERRORS, MAP RELOCATION DISABLED?
5397  033500  020123  042101  051104
5398  033506  051505  044523  043516
5399  033514  042440  051122  051117
5400  033522  026123  046440  050101
5401  033530  051040  046105  041517
5402  033536  052101  047511  020116
5403  033544  044504  040523  046102
5404  033552  042105     000
5405  033555     103  052517  052116  EM207:  .ASCIZ  ?COUNT PATTERN FAILURES IN CACHE REGISTERS?
5406  033562  050040  052101  042524
5407  033570  047122  043040  044501
5408  033576  052514  042522  020123
5409  033604  047111  041440  041501
5410  033612  042510  051040  043505
5411  033620  051511  042524  051522
5412  033626     000
5413
5414  033627     122  041505  044505  DH1:    .ASCIZ  ?RECEIVD EXPECTD TESTNO  PC AT ABORT?
5415  033634  042126  042440  050130
5416  033642  041505  042124  052040
5417  033650  051505  047124  020117
5418  033656  050040  020103  052101
5419  033664  040440  047502  052122
5420  033672     000
5421  033673     122  041505  044505  DH2:    .ASCIZ  ?RECEIVD TESTNO  PC AT ABORT?
5422  033700  042126  052040  051505
5423  033706  047124  020117  050040
5424  033714  020103  052101  040440
5425  033722  047502  052122     000
5426  033727     103  047117  044504  DH3:    .ASCII  ?CONDITN ADDRESS   MAINTEN CONTROL?<CRLF>
5427  033734  047124  040440  042104
5428  033742  042522  051523  020040
5429  033750  046440  044501  052116
5430  033756  047105  041440  047117
5431  033764  051124  046117     200
5432  033771     122  041505  044505          .ASCIZ  ?RECEIVD REFERENCD REGISTR REGISTR TESTNO  PC AT ABORT?
5433  033776  042126  051040  043105
5434  034004  051105  047105  042103
5435  034012  051040  043505  051511
5436  034020  051124  051040  043505
```

```
 5437  034026  051511  051124  052040
 5438  034034  051505  047124  020117
 5439  034042  050040  020103  052101
 5440  034050  040440  047502  052122
 5441  034056     000
 5442  034057     123  040524  052524  DH5:    .ASCII  ?STATUS  AUTOI/D VIRTUAL?<CRLF>
 5443  034064  020123  040440  052125
 5444  034072  044517  042057  053040
 5445  034100  051111  052524  046101
 5446  034106     200
 5447  034107     122  043505  051511          .ASCIZ  ?REGISTR REGISTR ADDRESS TESTNO  PC AT ABORT?
 5448  034114  051124  051040  043505
 5449  034122  051511  051124  040440
 5450  034130  042104  042522  051523
 5451  034136  052040  051505  047124
 5452  034144  020117  050040  020103
 5453  034152  052101  040440  047502
 5454  034160  052122     000
 5455  034163     122  043505  042101  DH6:    .ASCII  ?REGADRS REGADRS?<CRLF>
 5456  034170  051522  051040  043505
 5457  034176  042101  051522     200
 5458  034203     040  047442  021122          .ASCIZ  ? ''OR''   ''AND''  #ERRORS TESTNO  ERRORPC?
 5459  034210  020040  020040  040442
 5460  034216  042116  020042  021440
 5461  034224  051105  047522  051522
 5462  034232  052040  051505  047124
 5463  034240  020117  042440  051122
 5464  034246  051117  041520     000
 5465  034253     122  043505  042101  DH10:   .ASCII  ?REGADRS REGADRS RECEIVD RECEIVD?<CRLF>
 5466  034260  051522  051040  043505
 5467  034266  042101  051522  051040
 5468  034274  041505  044505  042126
 5469  034302  051040  041505  044505
 5470  034310  042126     200
 5471  034313     040  047442  021122          .ASCIZ  ? ''OR''   ''AND''  ''OR''   ''AND''  #ERRORS TESTNO  ERRORPC?
 5472  034320  020040  020040  040442
 5473  034326  042116  020042  020040
 5474  034334  047442  021122  020040
 5475  034342  020040  040442  042116
 5476  034350  020042  021440  051105
 5477  034356  047522  051522  052040
 5478  034364  051505  047124  020117
 5479  034372  042440  051122  051117
 5480  034400  041520     000
 5481  034403     103  052517  052116  DH12:   .ASCII  ?COUNT   COUNT?<CRLF>
 5482  034410  020040  041440  052517
 5483  034416  052116     200
 5484  034421     105  050130  041505          .ASCIZ  ?EXPECTD RECEIVD TESTNO  ERRORPC?
 5485  034426  042124  051040  041505
 5486  034434  044505  042126  052040
 5487  034442  051505  047124  020117
 5488  034450  042440  051122  051117
 5489  034456  041520     000
 5490  034461     122  043505  047514  DH14:   .ASCII  ?REGLOAD REGLOAD REGDUAL REGDUAL?<CRLF>
 5491  034466  042101  051040  043505
 5492  034474  047514  042101  051040
```

```
5493  034502  043505  052504  046101
5494  034510  051040  043505  052504
5495  034516  046101     200
5496  034521     040  047442  021122          .ASCIZ  ? 'DR''   ''AND''   'DR''   ''AND''  #ERRORS TESTNO?
5497  034526  020040  020040  040442
5498  034534  042116  020042  020040
5499  034542  047442  021122  020040
5500  034550  020040  040442  042116
5501  034556  020042  021440  051105
5502  034564  047522  051522  052040
5503  034572  051505  047124  000117
5504  034600  040515  051120  043505  DH15:   .ASCII  ?MAPREG  MAPREG  EXPECTD EXPECTD RECEIVD RECEIVD?<CRLF>
5505  034606  020040  040515  051120
5506  034614  043505  020040  054105
5507  034622  042520  052103  020104
5508  034630  054105  042520  052103
5509  034636  020104  042522  042503
5510  034644  053111  020104  042522
5511  034652  042503  053111  100104
5512  034660  021040  051117  020042          .ASCIZ  ? 'DR''   ''AND''   'DR''   ''AND''   'DR''   ''AND''  #ERRORS TESTNO?
5513  034666  020040  021040  047101
5514  034674  021104  020040  021040
5515  034702  051117  020042  020040
5516  034710  021040  047101  021104
5517  034716  020040  021040  051117
5518  034724  020042  020040  021040
5519  034732  047101  021104  020040
5520  034740  042443  051122  051117
5521  034746  020123  042524  052123
5522  034754  047516     000
5523  034757     122  041505  044505  DH17:   .ASCIZ  ?RECEIVD TESTNO  ERRORPC?
5524  034764  042126  052040  051505
5525  034772  047124  020117  042440
5526  035000  051122  051117  041520
5527  035006     000
5528  035007     105  050130  041505  DH23:   .ASCII  ?EXPECTD EXPECTD RECEIVD RECEIVD?<CRLF>
5529  035014  042124  042440  050130
5530  035022  041505  042124  051040
5531  035030  041505  044505  042126
5532  035036  051040  041505  044505
5533  035044  042126     200
5534  035047     040  047442  021122          .ASCIZ  ? 'DR''   ''AND''   'DR''   ''AND''  #ERRORS TESTNO?
5535  035054  020040  020040  040442
5536  035062  042116  020042  020040
5537  035070  047442  021122  020040
5538  035076  020040  040442  042116
5539  035104  020042  021440  051105
5540  035112  047522  051522  052040
5541  035120  051505  047124  000117
5542  035126  042101  051104  051525  DH25:   .ASCIZ  ?ADDRUSED  BITDIFF TESTNO  ERRORPC?
5543  035134  042105  020040  044502
5544  035142  042124  043111  020106
5545  035150  042524  052123  047516
5546  035156  020040  051105  047522
5547  035164  050122  000103
5548  035170  042101  051104  051525  DH27:   .ASCIZ  ?ADDRUSED  TESTNO  ERRORPC?
```

```
5549  035176  042105  020040  042524
5550  035204  052123  047516  020040
5551  035212  051105  047522  050122
5552  035220  000103
5553  035222  042524  052123  047516  DH30:   .ASCIZ  ?TESTNO  ERRORPC?
5554  035230  020040  051105  047522
5555  035236  050122  000103
5556  035242  042522  047515  042526  DH34:   .ASCIZ  ?REMOVED MISSING TESTNO  ERRORPC?
5557  035250  020104  044515  051523
5558  035256  047111  020107  042524
5559  035264  052123  047516  020040
5560  035272  051105  047522  050122
5561  035300  000103
5562  035302  047514  042527  052123  DH35:   .ASCIZ  ?LOWEST  HIGHEST TESTNO  ERRORPC?
5563  035310  020040  044510  044107
5564  035316  051505  020124  042524
5565  035324  052123  047516  020040
5566  035332  051105  047522  050122
5567  035340  000103
5568  035342  042524  052123  047516  DH36:   .ASCIZ  ?TESTNO  ERRORPC UNIBUS ADDRESS OF MAP REGISTER UNDER TEST?
5569  035350  020040  051105  047522
5570  035356  050122  020103  047125
5571  035364  041111  051525  040440
5572  035372  042104  042522  051523
5573  035400  047440  020106  040515
5574  035406  020120  042522  044507
5575  035414  052123  051105  052440
5576  035422  042116  051105  052040
5577  035430  051505  000124
5578  035434  047503  042116  052111  DH40:   .ASCII  ?CONDITN CONDITN?<CRLF>
5579  035442  020116  047503  042116
5580  035450  052111  100116
5581  035454  054105  042520  052103          .ASCIZ  ?EXPECTD RECEIVD TESTNO  ERRORPC?
5582  035462  020104  042522  042503
5583  035470  053111  020104  042524
5584  035476  052123  047516  020040
5585  035504  051105  047522  050122
5586  035512  000103
5587  035514  047503  051122  041505  DH41:   .ASCII  ?CORRECT ADDRESS?<CRLF>
5588  035522  020124  042101  051104
5589  035530  051505  100123
5590  035534  042101  051104  051505          .ASCIZ  ?ADDRESS FETCHED TESTNO  ERRORPC?
5591  035542  020123  042506  041524
5592  035550  042510  020104  042524
5593  035556  052123  047516  020040
5594  035564  051105  047522  050122
5595  035572  000103
5596  035574  047503  051122  041505  DH42:   .ASCII  ?CORRECT EXPECTD RECEIVD?<CRLF>
5597  035602  020124  054105  042520
5598  035610  052103  020104  042522
5599  035616  042503  053111  100104
5600  035624  042101  051104  051505          .ASCIZ  ?ADDRESS DATA    FROM UB TESTNO  ERRORPC?
5601  035632  020123  040504  040524
5602  035640  020040  020040  051106
5603  035646  046517  052440  020102
5604  035654  042524  052123  047516
```

```
5605  035662  020040  051105  047522
5606  035670  050122  000103
5607  035674  044523  045132  046525  DH43:   .ASCIZ  ?SIZJUMP TOPFOUND TESTNO ERRORPC?
5608  035702  020120  047524  043120
5609  035710  052517  042116  052040
5610  035716  051505  047124  020117
5611  035724  051105  047522  050122
5612  035732  000103
5613  035734  047503  042116  052111  DH44:   .ASCII  ?CONDITN CONDITN ADDRESS   MAINTEN CONTROL?<CRLF>
5614  035742  020116  047503  042116
5615  035750  052111  020116  042101
5616  035756  051104  051505  020123
5617  035764  020040  040515  047111
5618  035772  042524  020116  047503
5619  036000  052116  047522  100114
5620  036006  054105  042520  052103          .ASCIZ  ?EXPECTD RECEIVD REFERENCD REGISTR REGISTR TESTNO  ERRORPC?
5621  036014  020104  042522  042503
5622  036022  053111  020104  042522
5623  036030  042506  042522  041516
5624  036036  020104  042522  044507
5625  036044  052123  020122  042522
5626  036052  044507  052123  020122
5627  036060  042524  052123  047516
5628  036066  020040  051105  047522
5629  036074  050122  000103
5630  036100  020040  020040  041520  DH53:   .ASCIZ  /    PC    UBMR  (UBMR)/
5631  036106  020040  020040  041125
5632  036114  051115  020040  052450
5633  036122  046502  024522     000
5634  036127     105  051122  051117  DH55:   .ASCIZ  /ERRORPC  MKCSR     MSER     CPUERR/
5635  036134  041520  020040  045515
5636  036142  051503  020122  020040
5637  036150  020040  051515  051105
5638  036156  020040  020040  041440
5639  036164  052520  051105  000122
5640  036172  020040  041520  020040  DH56:   .ASCIZ  /  PC     CCR   GROUP   TST-DATA-ADRS/
5641  036200  020040  041440  051103
5642  036206  020040  043440  047522
5643  036214  050125  020040  052040
5644  036222  052123  042055  052101
5645  036230  026501  042101  051522
5646  036236     000
5647  036237     040  050040  020103  DH57:   .ASCIZ  /  PC    CCR    UBMR   TST-DATA-ADRS/
5648  036244  020040  041440  051103
5649  036252  020040  020040  041125
5650  036260  051115  020040  052040
5651  036266  052123  042055  052101
5652  036274  026501  042101  051522
5653  036302     000
5654  036303     122  043505  042101  DH201:  .ASCIZ  ?REGADRS TESTNO  ERRORPC?
5655  036310  051522  052040  051505
5656  036316  047124  020117  042440
5657  036324  051122  051117  041520
5658  036332     000
5659  036333     122  043505  042101  DH202:  .ASCIZ  ?REGADRS DATAREC TESTNO  ERRORPC?
5660  036340  051522  042040  052101
```

```
5661  036346  051101  041505  052040
5662  036354  051505  047124  020117
5663  036362  042440  051122  051117
5664  036370  041520     000
5665  036373     115  050101  042522  DH203:  .ASCII  ?MAPREG  MAPREG?<CRLF>
5666  036400  020107  046440  050101
5667  036406  042522  100107
5668  036412  042524  052123  047111          .ASCIZ  ?TESTING DUALED  TESTNO  ERRORPC?
5669  036420  020107  052504  046101
5670  036426  042105  020040  042524
5671  036434  052123  047516  020040
5672  036442  051105  047522  050122
5673  036450  000103
5674  036452  042522  040507  051104  DH204:  .ASCIZ  ?REGADRS PATTERN EXPECTD RECEIVD TESTNO  ERRORPC?
5675  036460  020123  040520  052124
5676  036466  051105  020116  054105
5677  036474  042520  052103  020104
5678  036502  042522  042503  053111
5679  036510  020104  042524  052123
5680  036516  047516  020040  051105
5681  036524  047522  050122  000103
5682  036532  054105  042520  052103  DH205:  .ASCIZ  ?EXPECTD RECEIVD ADDRSLOAD TESTNO  ERRORPC?
5683  036540  020104  042522  042503
5684  034546  053111  020104  042101
5685  036554  051104  046123  040515
5686  036562  020104  042524  052123
5687  036570  045516  020040  051105
5688  036576  047522  050122  000103
5689  036604  042101  051104  051505  DH206:  .ASCII  ?ADDRESS ADDRESS?<CRLF>
5690  036612  020123  042101  051104
5691  036620  051505  100123
5692  036624  054105  042520  052103          .ASCIZ  ?EXPECTD RECEIVD TESTNO  ERRORPC?
5693  036632  020104  042522  042503
5694  036640  053111  020104  042524
5695  036646  052123  047516  020040
5696  036654  051105  047522  050122
5697  036662  000103
5698  036664  042522  044507  052123  DH207:  .ASCII  ?REGISTR EXPECTD RECEIVD?<CRLF>
5699  036672  020122  054105  042520
5700  036700  052103  020104  042522
5701  036706  042503  053111  100104
5702  036714  042101  051104  051505          .ASCIZ  ?ADDRESS  DATA    DATA   TESTNO  ERRORPC?
5703  036722  020123  042040  052101
5704  036730  020101  020040  042040
5705  036736  052101  020101  020040
5706  036744  042524  052123  047516
5707  036752  020040  051105  047522
5708  036760  050122  000103
5709
5710
5711                                          .EVEN
5712  036764  001266  001264  001262  DT1:    .WORD   PCPUER,CPUEXP,TESTNO,BADPC,0
5713  036772  001302  000000
5714  036776  001266  001262  001302  DT2:    .WORD   PCPUER,TESTNO,BADPC,0
5715  037004  000000
5716  037006  001274  001270  001300  DT3:    .WORD   PPARER,PLOADR,PMAINT,PCONTR,TESTNO,BADPC,0
```

```
5717  037014  001276  001262  001302
5718  037022  000000
5719  037024  001154  001170  001262  DT4:    .WORD   $REG0,$TMP0,TESTNO,$ERRPC,0
5720  037032  001116  000000
5721  037036  001312  001314  001316  DT5:    .WORD   PMMR0,PMMR1,PMMR2,TESTNO,BADPC,0
5722  037044  001262  001302  000000
5723  037052  001226  001224  001254  DT6:    .WORD   ADDROR,ADRAND,ERRCNT,TESTNO,$ERRPC,0
5724  037060  001262  001116  000000
5725  037066  001226  001224  001232  DT10:   .WORD   ADDROR,ADRAND,DATAOR,DATAND,ERRCNT,TESTNO,$ERRPC,0
5726  037074  001230  001254  001262
5727  037102  001116  000000
5728  037106  001160  001154  001262  DT12:   .WORD   $REG2,$REG0,TESTNO,$ERRPC,0
5729  037114  001116  000000
5730  037120  001162  001156  001262  DT13:   .WORD   $REG3,$REG1,TESTNO,$ERRPC,0
5731  037126  001116  000000
5732  037132  001226  001224  001232  DT14:   .WORD   ADDROR,ADRAND,DATAOR,DATAND,ERRCNT,TESTNO,0
5733  037140  001230  001254  001262
5734  037146  000000
5735  037150  001226  001224  001236  DT15:   .WORD   ADDROR,ADRAND,PATTOR,PATAND,DATAOR,DATAND,ERRCNT,TESTNO,0
5736  037156  001234  001232  001230
5737  037164  001254  001262  000000
5738  037172  001154  001262  001116  DT17:   .WORD   $REG0,TESTNO,$ERRPC,0
5739  037200  000000
5740  037202  001156  001262  001116  DT20:   .WORD   $REG1,TESTNO,$ERRPC,0
5741  037210  000000
5742  037212  001236  001234  001232  DT23:   .WORD   PATTOR,PATAND,DATAOR,DATAND,ERRCNT,TESTNO,0
5743  037220  001230  001254  001262
5744  037226  000000
5745  037230  001164  001154  001262  DT25:   .WORD   $REG4,$REG0,TESTNO,$ERRPC,0
5746  037236  001116  000000
5747  037242  001156  001262  001116  DT27:   .WORD   $REG1,TESTNO,$ERRPC,0
5748  037250  000000
5749  037252  001262  001116  000000  DT30:   .WORD   TESTNO,$ERRPC,0
5750  037260  001254  001256  001262  DT34:   .WORD   ERRCNT,CNTR,TESTNO,$ERRPC,0
5751  037266  001116  000000
5752  037272  001240  001242  001262  DT35:   .WORD   LOWEST,HIGEST,TESTNO,$ERRPC,0
5753  037300  001116  000000
5754  037304  001262  001116  001154  DT36:   .WORD   TESTNO,$ERRPC,$REG0,0
5755  037312  000000
5756  037314  001226  001224  001232  DT37:   .WORD   ADDROR,ADRAND,DATAOR,DATAND,ERRCNT,TESTNO,0
5757  037322  001230  001254  001262
5758  037330  000000
5759  037332  001264  001266  001262  DT40:   .WORD   CPUEXP,PCPUER,TESTNO,$ERRPC,0
5760  037340  001116  000000
5761  037344  001160  001156  001262  DT41:   .WORD   $REG2,$REG1,TESTNO,$ERRPC,0
5762  037352  001116  000000
5763  037356  001156  001162  001160  DT42:   .WORD   $REG1,$REG3,$REG2,TESTNO,$ERRPC,0
5764  037364  001262  001116  000000
5765  037372  177760  001320  001262  DT43:   .WORD   SIZELO,RSIZE,TESTNO,$ERRPC,0
5766  037400  001116  000000
5767  037404  001200  001274  001270  DT44:   .WORD   $TMP4,PPARER,PLOADR,PMAINT,PCONTR,TESTNO,$ERRPC,0
5768  037412  001300  001276  001262
5769  037420  001116  000000
5770  037424  001116  001154  001156  DT53:   .WORD   $ERRPC,$REG0,$REG1,0
5771  037432  000000
5772  037434  001262  001160  001172  DT55:   .WORD   TESTNO,$REG2,$TMP1,$TMP2,0
```

```
5773  037442  001174  000000
5774  037446  001116  001154  001156  DT56:   .WORD   $ERRPC,$REG0,$REG1,$REG2,0
5775  037454  001160  000000
5776  037460  001116  001154  001156  DT57:   .WORD   $ERRPC,$REG0,$REG1,$REG2,0
5777  037466  001160  000000
5778  037472  001154  001262  001116  DT201:  .WORD   $REG0,TESTNO,$ERRPC,0
5779  037500  000000
5780  037502  001154  001170  001262  DT202:  .WORD   $REG0,$TMP0,TESTNO,$ERRPC,0
5781  037510  001116  000000
5782  037514  001154  001156  001262  DT203:  .WORD   $REG0,$REG1,TESTNO,$ERRPC,0
5783  037522  001116  000000
5784  037526  001154  001160  001164  DT204:  .WORD   $REG0,$REG2,$REG4,$REG3,TESTNO,$ERRPC,0
5785  037534  001162  001262  001116
5786  037542  000000
5787  037544  001156  001154  001160  DT205:  .WORD   $REG1,$REG0,$REG2,TESTNO,$ERRPC,0
5788  037552  001262  001116  000000
5789  037560  001154  001162  001262  DT206:  .WORD   $REG0,$REG3,TESTNO,$ERRPC,0
5790  037566  001116  000000
5791  037572  001154  001160  001162  DT207:  .WORD   $REG0,$REG2,$REG3,TESTNO,$ERRPC,0
5792  037600  001262  001116  000000
5793
5794
5795  037606      000     000     000  DF1:    .BYTE   0,0,0,0
5796  037611      000
5797  037612      000     000     000  DF2:    .BYTE   0,0,0
5798  037615      000     002     000  DF3:    .BYTE   0,2,0,0,0,0
5799  037620      000     000     000
5800  037623      000     000     000  DF5:    .BYTE   0,0,0,0,0
5801  037626      000     000
5802  037630      000     000     001  DF6:    .BYTE   0,0,1,0,0
5803  037633      000     000
5804  037635      000     000     000  DF10:   .BYTE   0,0,0,0,1,0,0
5805  037640      000     001     000
5806  037643      000
5807  037644      000     000     000  DF12:   .BYTE   0,0,0,0
5808  037647      000
5809  037650      000     000     000  DF14:   .BYTE   0,0,0,0,1,0
5810  037653      000     001     000
5811  037656      000     000     000  DF15:   .BYTE   0,0,0,0,0,0,1,0
5812  037661      000     000     000
5813  037664      001     000
5814  037666      000     000     000  DF17:   .BYTE   0,0,0
5815  037671      000     000     000  DF23:   .BYTE   0,0,0,0,1,0
5816  037674      000     001     000
5817  037677      003     004     000  DF25:   .BYTE   3,4,0,0
5818  037702      000
5819  037703      003     000     000  DF27:   .BYTE   3,0,0
5820  037706      000     000          DF30:   .BYTE   0,0
5821  037710      000     000     000  DF34:   .BYTE   0,0,0,0
5822  037713      000
5823  037714      004     004     000  DF35:   .BYTE   4,4,0,0
5824  037717      000
5825  037720      000     000     003  DF36:   .BYTE   0,0,3
5826  037723      000     000     000  DF37:   .BYTE   0,0,0,0,1,0
5827  037726      000     001     000
5828  037731      000     000     000  DF40:   .BYTE   0,0,0,0
```

```
5829  037734  000
5830  037735  000  000  000  DF41:  .BYTE  0,0,0,0
5831  037740  000
5832  037741  003  000  000  DF42:  .BYTE  3,0,0,0,0
5833  037744  000  000
5834  037746  000  000  000  DF43:  .BYTE  0,0,0,0
5835  037751  000
5836  037752  000  000  002  DF44:  .BYTE  0,0,2,0,0,0,0
5837  037755  000  000  000
5838  037760  000
5839  037761  000  000  000  DF201:  .BYTE  0,0,0
5840  037764  000  000  000  DF202:  .BYTE  0,0,0,0
5841  037767  000
5842  037770  000  000  000  DF203:  .BYTE  0,0,0,0
5843  037773  000
5844  037774  000  000  000  DF204:  .BYTE  0,0,0,0,0,0
5845  037777  000  000  000
5846  040002  000  000  003  DF205:  .BYTE  0,0,3,0,0
5847  040005  000  000
5848  040007  000  000  000  DF206:  .BYTE  0,0,0,0
5849  040012  000
5850  040013  000  000  000  DF207:  .BYTE  0,0,0,0,0
5851  040016  000  000
5852
5853                                .EVEN
5854  040020  000017              SAVEA:  .BLKW  17
5855
5856          040056              TSLOC=.                    ;GET PC TO AN EVEN WORD BOUNDARY
5857          040054              TSLOC=-4&TSLOC
5858          040060              TSLOC=TSLOC+4
5859          040060              .=TSLOC
5860
5861  040060  001000              TSTDAT:  .BLKW  512.
5862          000001                       .END
```

```
ADDROR  001226  518#  986*  1754*  1780*  1807*  1833*  2854*  3322*  5723  5725  5732  5735  5756
ADRAND  001224  517#  990*  1756*  1782*  1809*  1835*  2858*  3317*  5723  5725  5732  5735  5756
BADPC   001302  546#  1938*  1984*  2030*  5712  5714  5716  5721
BIT0  = 000001  132#  2120  2121  2128  2908
BIT00 = 000001  122#  132
BIT01 = 000002  121#  131
BIT02 = 000004  120#  130
BIT03 = 000010  119#  129
BIT04 = 000020  118#  128
BIT05 = 000040  117#  127
BIT06 = 000100  116#  126
BIT07 = 000200  115#  125
BIT08 = 000400  114#  124  1006
BIT09 = 001000  113#  123  1016  1089
BIT1  = 000002  131#  1993*
BIT10 = 002000  112#  1072
BIT11 = 004000  111#  1023  2123  2124  2126  2934
BIT12 = 010000  110#  416  4759
BIT13 = 020000  109#  1079
BIT14 = 040000  108#  2115  2116  2118
BIT15 = 100000  107#  415  2129  2130  2132  2134  2135  2137  3778  3849  4378  4451
BIT2  = 000004  130#  419  420  3767  4367
BIT3  = 000010  129#  419  420  3838  4440
BIT4  = 000020  128#  418  420  1685  2909  2928  4558
BIT5  = 000040  127#  417  419  2973  3143  3279  3354  3391  3575
BIT6  = 000100  126#
BIT7  = 000200  125#
BIT8  = 000400  124#  2150
BIT9  = 001000  123#  1142
BPTVEC= 000014  139#
BYP   = 100000  415#  4504  4510  4511  4517  4518  4654  4668
CACHVE= 000114  146#  1975*  1987*  19??*  2176*  2178*  3665*  3676*  3722*  3733*  3785*  3804*  3856*
                3875*  4259*  4270*  43??*  4330*  4385*  4404*  4458*  4477*
CACOUN  005576  1891#  2843  2873
CBMP    023714  4588  4590  4592#  4601
CBMPA   023734  4597#  4696
CISP    001333  561#
CLRMAP  005134  1722#  2546  3032  3228
CNTR    001256  536#  3171*  3175  3179*  5750
CONTRL= 177746  156#  1982  2112  2142*  2143  2773  2777*  2778  2832  2833  2835  2860*  2880*
                3656  3657*  3677*  3685  3711  3712*  3734*  3742  3767  3769  3782*  3805*  3813
                3838  3840  3853*  3876*  3884  4250  4251*  4271*  4279  4308  4309*  4331*  4339
                4367  4369  4382*  4405*  4413  4440  4442  4455*  4478*  4486  4597*  4598  4603*
                4605*  4611*  4641  4676
COUNT   005432  1832#  2599  2643  2688  2732
CPFLAG  005654  1102*  1926#  1947*  2192*
CPUER   005652  1925#  2182  2290
CPUERR  177766  169#  1098*  1748  1749*  1937  1946*  2197*  4558  4562  4564*
CPUEXP  001264  539#  1750  1939  1941  2188*  2929*  2952*  2969*  3007*  3033*  3071*  3092*  3110*
                3137*  3180*  3355*  3373*  3585*  3596*  3630*  3945*  3963*  4179*  4190*  4224*  5712
                5759
CR      000015  44#  1392  1402
CRLF  - 000200  45#  1365  1402  2090  4775  4781  4783  4785  4790  4937  4952  4968  4986
                5038  5052  5090  5109  5127  5136  5146  5229  5245  5260  5276  5291  5426
                5442  5455  5465  5481  5490  5504  5528  5578  5587  5596  5613  5665  5689
                5698
```

DATA 001326 557# 3587 3611* 3618* 3619* 3620* 4181 4205* 4212* 4213* 4214* 5735 5742
DATAND 001230 519# 989* 1786* 1813* 1839* 1870* 1899* 2857* 3320* 5725 5732 5735 5742
5756
DATAOR 001232 520# 985* 1784* 1811* 1837* 1868* 1897* 2853* 3323* 5725 5732 5735 5742
5756
DF1 037606 586 899 913 5795#
DF10 037635 636 643 5804#
DF12 037644 650 657 906 5807#
DF14 037650 664 5809#
DF15 037656 671 678 5811#
DF17 037666 685 692 699 706 5814#
DF2 037612 592 5797#
DF201 037761 923 5839#
DF202 037764 929 5840#
DF203 037770 936 5842#
DF204 037774 942 5844#
DF205 040002 948 5846#
DF206 040007 955 5848#
DF207 040013 962 5850#
DF23 037671 713 720 755 5815#
DF25 037677 727 734 5817#
DF27 037703 741 5819#
DF3 037615 600 608 5798#
DF30 037706 748 761 769 886 893 5820#
DF34 037710 776 5821#
DF35 037714 785 5823#
DF36 037720 791 5825#
DF37 037723 798 879 5826#
DF40 037731 805 840 5828#
DF41 037735 812 848 5830#
DF42 037741 819 856 5832#
DF43 037746 825 863 5834#
DF44 037752 832 872 5836#
DF5 037623 615 5800#
DF6 037630 622 629 5802#
DH1 033627 584 5414#
DH10 034253 633 640 5465#
DH12 034403 647 654 5481#
DH14 034461 661 5490#
DH15 034600 668 675 5504#
DH17 034757 683 690 697 704 5523#
DH2 033673 590 5421#
DH201 036303 921 5654#
DH202 036333 927 5659#
DH203 036373 933 5665#
DH204 036452 940 5674#
DH205 036532 946 5682#
DH206 036604 952 5689#
DH207 036664 959 5698#
DH23 035007 710 717 752 795 876 5528#
DH25 035126 725 732 5542#
DH27 035170 739 5548#
DH3 033727 597 605 5426#
DH30 035222 746 759 767 5553#
DH34 035242 774 5556#
DH35 035302 783 5562#

DH36	035342	789	5568#				
DH40	035434	802	837	5578#			
DH41	035514	809	845	5587#			
DH42	035574	816	853	5596#			
DH43	035674	823	861	5607#			
DH44	035734	829	869	5613#			
DH5	034057	612	5442#				
DH53	036100	884	891	5630#			
DH55	036127	897	5634#				
DH56	036172	904	5640#				
DH57	036237	911	5647#				
DH6	034163	619	626	5455#			
DISPLA=	177570	39#	1037*	1068*	2927*	3136*	3940*
DT1	036764	585	5712#				
DT1C	037066	635	642	5725#			
DT12	037106	649	5728#				
DT13	037120	656	5730#				
DT14	037132	663	5732#				
DT15	037150	670	677	5735#			
DT17	037172	684	698	705	5738#		
DT2	036776	591	5714#				
DT20	037202	691	5740#				
DT201	037472	922	5778#				
DT202	037502	928	5780#				
DT203	037514	935	5782#				
DT204	037526	941	5784#				
DT205	037544	947	5787#				
DT206	037560	954	5789#				
DT207	037572	961	5791#				
DT23	037212	712	719	754	5742#		
DT25	037230	726	733	5745#			
DT27	037242	740	5747#				
DT3	037006	599	607	5716#			
DT30	037252	747	760	768	5749#		
DT34	037260	775	5750#				
DT35	037272	784	5752#				
DT36	037304	790	5754#				
DT37	037314	797	878	5756#			
DT4	037024	5719#					
DT40	037332	804	839	5759#			
DT41	037344	811	847	5761#			
DT42	037356	818	855	5763#			
DT43	037372	824	862	5765#			
DT44	037404	831	871	5767#			
DT5	037036	614	5721#				
DT53	037424	885	892	5770#			
DT55	037434	898	5772#				
DT56	037446	905	5774#				
DT57	037460	912	5776#				
DT6	037052	621	628	5723#			
DUALAD	005356	1805#	2554	3244			
EMTVEC=	000030	142#	2059*	2060*			
EM1	025005	583	4795#				
EM10	025616	632	4863#				
EM11	025707	639	4873#				
EM12	026001	646	4883#				

EM13	026067	653	4893#											
EM14	026161	660	4903#											
EM15	026254	667	4913#											
EM16	026360	674	4925#											
EM17	026463	681	4937#											
EM2	025067	589	4804#											
EM20	026605	688	4952#											
EM201	033122	920	5355#											
EM202	033176	926	5363#											
EM203	033251	932	5371#											
EM204	033344	939	5381#											
EM205	033424	945	5389#											
EM206	033473	951	5396#											
EM207	033555	958	5405#											
EM21	026733	695	4968#											
EM22	027077	702	4986#											
EM23	027243	709	5004#											
EM24	027337	716	5015#											
EM25	027437	723	5026#											
EM26	027544	730	737	5038#										
EM3	025136	595	4811#											
EM30	027665	744	5052#											
EM31	030041	751	5072#											
EM32	030133	758	5082#											
EM33	030205	764	5090#											
EM34	030362	772	5109#											
EM35	030533	779	5127#											
EM36	031020	788	5160#											
EM37	031115	794	5171#											
EM4	025237	603	4822#											
EM40	031220	801	5183#											
EM41	031313	808	5193#											
EM42	031375	815	5202#											
EM43	031470	822	5212#											
EM44	031555	828	5221#											
EM45	031632	835	5229#											
EM46	031765	843	5245#											
EM47	032107	851	5260#											
EM5	025346	611	4834#											
EM50	032242	859	5276#											
EM51	032367	867	5291#											
EM52	032504	875	5305#											
EM53	032543	883	5311#											
EM54	032610	890	5318#											
EM55	032651	896	5324#											
EM56	032725	903	5332#											
EM57	032763	910	5338#											
EM6	025454	618	4846#											
EM7	025534	625	4854#											
ENDKB	010570	2153	2158#											
EREXIT	003034	1097	1105#											
ERRCNT	001254	535#	984*	1059*	1758	1788	1815	1845	1872	2230	2261	2303	2337	2371
		2405	2563	2610	2654	2699	2743	2847	2852*	2876	3066	3167*	3175	3178*
		3259	3324	3330	5723	5725	5732	5735	5742	5750	5756			
ERRVEC=	000004	135#	997	998*	1000*	1003*	2182*	2183*	2220*	2290*				
ERTYPE	003036	1081	1132#											

```
ER200   002124          915#  1152    1155
FLAG    001260          537#
GNS   = ****** U        429   1573    1574    1575    1576    1577    1578    1579    1580    1581    2089    4728    4735
HIADRS= 177742          154#  1980    2799    3683    3740    3811    3882    4277    4337    4411    4484
HIGEST  001242          525#  3158*   3185    3192    3248    3257    5752
HITMIS= 177752          158#  2257    4639    4674
HREGL   001250          531#  3203*
HREGU   001252          533#  3205*
HT    = 000011           42#  1363    1402
IOTVEC= 000020          140#  2057*   2058*
KBTST   010314         2104#
KB11CM  001132          560#  2104*   2152*   2164    2486    3627    4221    4500    4542    4589
KB11E   001130          558#  2105*   2109*   2148    2150*   2162    2170
KB11EM  001331          559#  2484    3625    4219    4498    4540    4587
KDPAR0= 172360          312#
KDPAR1= 172362          313#
KDPAR2= 172364          314#
KDPAR3= 172366          315#
KDPAR4= 172370          316#
KDPAR5= 172372          317#
KDPAR6= 172374          318#
KDPAR7= 172376          319#
KDPDR0= 172320          290#
KDPDR1= 172322          291#
KDPDR2= 172324          292#
KDPDR3= 172326          293#
KDPDR4= 172330          294#
KDPDR5= 172332          295#
KDPDR6= 172334          296#
KDPDR7= 172336          297#
KERSTK= 001100           29#  1935
KIPAR0= 172340          301#  1244    2900*   3918*   4617*
KIPAR1= 172342          302#  2901*   3916*   4619*
KIPAR2= 172344          303#  2902*   4621*
KIPAR3= 172346          304#  2903*
KIPAR4= 172350          305#  2904*   3145*   3152*   3153    3157    3158    3159*   3160    3172*   3173    3238*   3280*
                       3287*  3288    3356*   3582*   3610    3612*   3946*   4176*   4204    4206*
KIPAR5= 172352          306#  2905*   3046*   3047*
KIPAR6= 172354          307#  2906*   2931*   2936*   2947*   2971*   2975*   2982*   3035*   3041*   3042    3046    3094
                       3295   3298    3310*   3311    3399*   3414*   3429*   3444*   3459*   3474*   3489*   3504*   3519*
                       3534*  3549*   3583*   3658*   3713*   3773*   3844*   3993*   4008*   4023*   4038*   4053*   4068*
                       4083*  4098*   4113*   4128*   4143*   4177*   4252*   4310*   4373*   4446*   4592*   4695*
KIPAR7= 172356          308#  2907*   4615*
KIPDR0= 172300          279#  2129*   2130    2132*   2892*   4618*
KIPDR1= 172302          280#  2893*   4620*
KIPDR2= 172304          281#  2894*   4622*
KIPDR3= 172306          282#  2895*
KIPDR4= 172310          283#  2896*
KIPDR5= 172312          284#  2897*
KIPDR6= 172314          285#  2898*   4593*
KIPDR7= 172316          286#  2899*   4616*
LF    = 000012           43#  1396    1402
LOADRS= 177740          153#  1979    2252    2792    3682    3739    3810    3881    4276    4336    4410    4483
LOOP    010652         2176#  4769
LOWEST  001240          523#  3157*   3183    3193    3233    3234    3356    3399    3414    3429    3444    3459    3474
                       3489   3504    3519    3534    3549    3583    3658    3713    3773    3844    3914    3946    3993
```

LREGL 001244
LREGU 001246
MAINT = 177750
MAPH0 = 170202
MAPH00= 170202
MAPH01= 170206
MAPH02= 170212
MAPH03= 170216
MAPH04= 170222
MAPH05= 170226
MAPH06= 170232
MAPH07= 170236
MAPH1 = 170206
MAPH10= 170242
MAPH11= 170246
MAPH12= 170252
MAPH13= 170256
MAPH14= 170262
MAPH15= 170266
MAPH16= 170272
MAPH17= 170276
MAPH2 = 170212
MAPH20= 170302
MAPH21= 170306
MAPH22= 170312
MAPH23= 170316
MAPH24= 170320
MAPH25= 170326
MAPH26= 170332
MAPH27= 170336
MAPH3 = 170216
MAPH30= 170342
MAPH31= 170346
MAPH32= 170352
MAPH33= 170356
MAPH34= 170362
MAPH35= 170366
MAPH36= 170372
MAPH37= 170376
MAPH4 = 170222
MAPH5 = 170226
MAPH6 = 170232
MAPH7 = 170236
MAPL0 = 170200
MAPL00= 170200
MAPL01= 170204
MAPL02= 170210
MAPL03= 170214

4008	4023	4038	4053	4068	4083	4098	4113	4128	4143	4177	4252	4310
4373	4446	5752										
527#	3200*	3232	3356*	3398*	3413*	3428*	3443*	3458*	3473*	3488*	3503*	3518*
3533*	3548*	3579*	3613*	3659*	3714*	3775*	3846*	3948*	3992*	4007*	4022*	4037*
4052*	4067*	4082*	4097*	4112*	4127*	4142*	4173*	4207*	4253*	4311*	4375*	4448*
529#	3202*	3357*	3397*	3412*	3427*	3442*	3457*	3472*	3487*	3502*	3517*	3532*
3547*	3578*	3615*	3616	3660*	3715*	3774*	3845*	3908	3947*	3991*	4006*	4021*
4036*	4051*	4066*	4081*	4096*	4111*	4126*	4141*	4172*	4209*	4210	4254*	4312*
4374*	4447*											
157#	1983	2110*	2113	2785*	2786	2861	3661	3666*	3716	3723*	3776	3786*
3847	3857*	4255	4260*	4313	4320*	4376	4386*	4449	4459*			
396#	2325	2492*										
332#	396	2114	2493	2637								
334#	398											
336#	400											
338#	402											
340#	404											
342#	406											
344#	408											
346#	410											
398#												
348#												
350#												
352#												
354#												
356#												
358#												
360#												
362#	2333	2646										
400#												
364#	2393	2506*	2507	2726								
366#												
368#												
370#												
372#												
374#												
376#												
378#												
402#												
380#												
382#												
384#												
386#												
388#												
390#	4691											
392#												
394#	1724	2226	2401	2545	2559	2735						
404#												
406#												
408#												
410#												
395#	1722	2221	2291	2543	2556	2933*	2942	3138				
331#	395	2441*	2442	2593	4595							
333#	397											
335#	399											
337#	401											

```
MAPL04= 170220          339#    403
MAPL05= 170224          341#    405
MAPL06= 170230          343#    407
MAPL07= 170234          345#    409
MAPL1  = 170204          397#
MAPL10= 170240          347#
MAPL11= 170244          349#
MAPL12= 170250          351#
MAPL13= 170254          353#
MAPL14= 170260          355#
MAPL15= 170264          357#
MAPL16= 170270          359#
MAPL17= 170274          361#   2299    2602
MAPL2  = 170210          399#
MAPL20= 170300          363#   2359    2453*   2454    2682
MAPL21= 170304          365#
MAPL22= 170310          367#
MAPL23= 170314          369#
MAPL24= 170320          371#
MAPL25= 170324          373#
MAPL26= 170330          375#
MAPL27= 170334          377#
MAPL3  = 170214          401#
MAPL30= 170340          379#
MAPL31= 170344          381#
MAPL32= 170350          383#
MAPL33= 170354          385#
MAPL34= 170360          387#
MAPL35= 170364          389#
MAPL36= 170370          391#
MAPL37= 170374          393#   2367    2691    3141
MAPL4  - 170220          403#
MAPL5    170224          405#
MAPL6  - 170230          407#
MAPL7  = 170234          409#
MEMER    005764        1967#   1987    1996    2176    3676    3733    3804    3875    4270    4330    4404    4477
MEMERR= 177744          155#   1099*   1981    2005*   2196*   2790*   2797*   3655*   3675*   3684    3710*   3732*   3741
                       3772*   3803*   3812    3843*   3874*   3883    4249*   4269*   4278    4307*   4329*   4338    4372*
                       4403*   4412    4445*   4476*   4485
MFPT   = 000007         564#   2107
MFPTYR  010562         2106    2155#
MKCSR  - 172100        4537#   4544
MMFLAG  006212         1104*   2023#   2038*   2195*
MMR0   = 177572         180#    184    1100*   2033    2037*   2189*   2908*   4636*   4666*   4713*
MMR1   = 177574         181#    185    2034
MMR2   = 177576         182#    186    2035
MMR3   = 172516         183#    187    2190*   2909*   3143*   3279*   3354*   3391*   4635*   4667*   4714*
MMTRAP  006210         2022#   2179
MMVEC  = 000250          148#   2179*   2181*
MSER   = 177744          421#   4539#   4556    4561
MSG1     024662        2161    4775#
MSG2     024721        2174    4781#
MSG3     024732        2166    4783#
MSG4     024744        2168    4785#
MSG5     024775        2172    4790#
NEXMEM= 000040        3575#   3596    3598    4190    4192
```

```
NXTTST  001324   555#  2003  2218* 2250* 2288* 2323* 2357* 2391* 2434* 2478* 2540* 2586* 2630*
                2675* 2719* 2769* 2828* 2967* 3029* 3090* 3128* 3225* 3277* 3349* 3389* 3572*
                3650* 3706* 3764* 3835* 3941* 3982* 4168* 4244* 4303* 4364* 4437* 4497* 4536*
                4586*
OLDPC   001304   547#  1746* 1765  1779* 1793  1806* 1820  1832* 1850  1863* 1877  1892* 1906
                1933* 1938  1949  1977* 1984  2001* 2003* 2008  2031* 2040
OLDPS   001306   548#  1747* 1764  1934* 1948  1978* 2007  2032* 2039
OLDPSW  001310   549#  1689* 1706  1707*
PADRSH  001222   515#  1250* 1272*
PADRSL  001220   513#  1251* 1252  1271* 1273
PAFLAG  005766  1103*  1968# 2006* 2194*
PATAND  001234   521#   991* 1843* 1866* 1895* 2859* 5735  5742
PATTOR  001236   522#   987* 1841* 1864* 1893* 2855* 5735  5742
PCONTR  001276   544#  1982* 3685* 3742* 3813* 3884* 4279* 4339* 4413* 4486* 5716  5767
PCPUER  001266   540#  1748* 1749  1750  1937* 1941  1946  3036* 3039  3360* 3369  3586* 3594
                3598  3950* 3959  4180* 4188  4192  5712  5714  5759
PHIADR  001272   542#  1980* 3683* 3740* 3811* 3882* 4277* 4337* 4411* 4484*
PIRQ  - 177772    37#
PIRQVE= 000240   147#
PLOADR  001270   541#  1979* 1991  3682* 3739* 3810* 3881* 4276* 4336* 4410* 4483* 5716  5767
PMAINT  001300   545#  1983* 3686* 3743* 3814* 3885* 4280* 4340* 4414* 4487* 5716  5767
PMMR0   001312   550#  2033* 5721
PMMR1   001314   551#  2034* 5721
PMMR2   001316   552#  2035* 5721
PPARER  001274   543#  1981* 1985  2005  3654* 3672  3684* 3709* 3729  3741* 3771* 3799  3812*
                3842* 3870  3883* 4248* 4266  4278* 4306* 4326  4338* 4371* 4399  4412* 4444*
                4472  4485* 5716  5767
PR0     = 000000   69#
PR1     = 000040   70#
PR2     = 000100   71#
PR3     = 000140   72#
PR4     = 000200   73#
PR5     = 000240   74#
PR6     = 000300   75#
PR7     = 000340   76#
PS      = 177776   34#    35   2051* 4580  4757
PSW     = 177776   35#
PWRMSG  004704  1615  1622#
PWRVEC= 000024   141#  1588* 1589* 1597* 1612* 1613* 2063* 2064*
RESREG= 104414  1259  1579#  1657
RESVEC= 000010   136#  2072* 2079* 2106*
RETRY   001322   554#   982* 1998  2000* 2850*
RSIZE   001320   553#  3610* 3622  4204* 4216  5765
SAVEA   040020  5854#
SAVREG= 104412  1236  1578#  1642
SDPAR0= 172260   268#
SDPAR1= 172262   269#
SDPAR2= 172264   270#
SDPAR3= 172266   271#
SDPAR4= 172270   272#
SDPAR5= 172272   273#
SDPAR6= 172274   274#
SDPAR7= 172276   275#
SDPDR0= 172220   246#
SDPDR1= 172222   247#
SDPDR2= 172224   248#
```

```
SDPDR3= 172226          249#
SDPDR4= 172230          250#
SDPDR5= 172232          251#
SDPDR6= 172234          252#
SDPDR7= 172236          253#
SIPAR0= 172240          257#
SIPAR1= 172242          258#
SIPAR2= 172244          259#
SIPAR3= 172246          260#
SIPAR4= 172250          261#
SIPAR5= 172252          262#
SIPAR6= 172254          263#
SIPAR7= 172256          264#
SIPDR0= 172200          235#
SIPDR1= 172202          236#
SIPDR2= 172204          237#
SIPDR3= 172206          238#
SIPDR4= 172210          239#
SIPDR5= 172212          240#
SIPDR6= 172214          241#
SIPDR7= 172216          242#
SIZEHI= 177762          166#
SIZEJ   014454         3045   3131#
SIZELO= 177760          164#  3351   3622   3942   4216   5765
SR0   = 177572          184#
SR1   = 177574          185#
SR2   = 177576          186#
SR3   = 172516          187#
STACK = 001100           28#    29     30     31   2056
START   010000          433   1617   2050#  3156
STKLMT= 177774           36#
SUPSTK= 000700           30#
SWR   = 177570           38#    39    992*  1006   1008   1016   1023   1069   1072   1079   1083   1089   1096
                        1145   4759
SW0   = 000001          104#
SW00  = 000001           94#   104
SW01  = 000002           93#   103
SW02  = 000004           92#   102
SW03  = 000010           91#   101
SW04  = 000020           90#   100
SW05  = 000040           89#    99
SW06  = 000100           88#    98
SW07  = 000200           87#    97
SW08  = 000400           86#    96
SW09  = 001000           85#    95
SW1     000002          103#
SW10  = 002000           84#
SW11  = 004000           83#
SW12  = 010000           82#
SW13  = 020000           81#
SW14  = 040000           80#
SW15  = 100000           79#
SW2   = 000004          102#
SW3   = 000010          101#
SW4   = 000020          100#
SW5   = 000040           99#
```

```
SW6    = 000100            98#
SW7    = 000200            97#  1145
SW8    = 000400            96#
SW9    = 001000            95#  1096
SYSTID= 177764            168#
S0     = 000020            418#
SOMOM1= 000034            420#  4605
S1     = 000040            417#  4611
S1MOM1= 000054            419#  4603
TBIT   = 000020           1685#  1687  1690  1707
TBITO  = 104416           1580#  2831  3652  4246
TBITOF  005070            1580  1686#
TBITR  = 104420           1581#  2888  3896  4715
TBITRE  005116            1581  1705#
TBITVE= 000014             137#  2069*  2070*
TESTNO  001262             538#  1058*  5712  5714  5716  5719  5721  5723  5725  5728  5730  5732  5735
                          5738  5740  5742  5745  5747  5749  5750  5752  5754  5756  5759  5761  5763
                          5765  5767  5772  5778  5780  5782  5784  5787  5789  5791

TIMEOU  005152            1737#  2220
TIMOUT= 000020            2928#  2929  2969  3033  3092  3137  3355  3369  4179
TKVEC  = 000060             144#
TOFLAG  005154            1101*  1739#  1763*  2193*
TPVEC  = 000064             145#
TRAPVE= 000034             143#  2061*  2062*
TRTVEC= 000014             138#
TSLOC  = 040060           5856#  5857#  5858#  5859
TSTDAT  040060            4609   4624   4629   4669  5861#
TST1    010774            2216#
TST10   011630            2434   2476#
TST11   012030            2478   2538#
TST12   012154            2540   2564   2584#
TST13   012300            2586   2611   2628#
TST14   012424            2630   2655   2673#
TST15   012550            2675   2700   2717#
TST16   012674            2719   2744   2767#
TST17   013072            2769   2826#
TST2    011060            2218   2231   2248#
TST20   013514            2828   2922#
TST21   013670            2965#
TST22   014072            2967   3027#
TST23   014324            3029   3088#
TST24   014436            3090   3126#
TST25   015074            3128   3182   3223#
TST26   015270            3225   3260   3275#
TST27   015550            3277   3297   3347#
TST3    011136            2250   2262   2286#
TST30   015676            3349   3353   3387#
TST31   016612            3389   3570#
TST32   017116            3572   3648#
TST33   017360            3650   3679   3704#
TST34   017632            3706   3736   3762#
TST35   020120            3764   3768   3807   3833#
TST36   020464            3835   3935#
TST37   020634            3941   3944   3980#
TST4    011234            2288   2304   2321#
TST40   021550            3982   4166#
```

```
TST41    022054            4168    4242#
TST42    022316            4244    4273    4301#
TST43    022570            4303    4333    4362#
TST44    023056            4364    4368    4407    4436#
TST45    023344            4437    4441    4480    4496#
TST46    023500            4497    4535#
TST47    023640            4536    4577#
TST5     011324            2323    2338    2355#
TST6     011414            2357    2372    2389#
TST7     011504            2391    2406    2432#
TYPDS  = 104410            1190    1577#   4732    4739
TYPE   = 104400            1074    1082    1162    1164    1169    1171    1199    1215    1256    1277    1368    1463    1539
                           1573#   1614    2087    2161    2166    2168    2172    2174    4726    4733    4740
TYPOC  = 104402            1137    1184    1574#
TYPON  = 104406            1576#
TYPOS  = 104404            1575#
TYPVAD   003344            1207    1236#
T.END    010510            2136    2139#
T1       010342            2110#   2156
T2       010404            2117    2120#
T3       010436            2122    2125    2128#
T4       010470            2131    2134#
UBADDR   003452            1212    1268#
UBCOUN   005522            1863#   3063
UBMAP    020406            3839    3878    3895#
UDPAR0=  177660             224#
UDPAR1=  177662             225#
UDPAR2=  177664             226#
UDPAR3=  177666             227#
UDPAR4=  177670             228#
UDPAR5=  177672             229#
UDPAR6=  177674             230#
UDPAR7=  177676             231#
UDPDR0=  177620             202#
UDPDR1=  177622             203#
UDPDR2=  177624             204#
UDPDR3=  177626             205#
UDPDR4=  177630             206#
UDPDR5=  177632             207#
UDPDR6=  177634             208#
UDPDR7=  177636             209#
UIPAR0=  177640             213#
UIPAR1=  177642             214#
UIPAR2=  177644             215#
UIPAR3=  177646             216#
UIPAR4=  177650             217#
UIPAR5=  177652             218#
UIPAR6=  177654             219#
UIPAR7=  177656             220#
UIPDR0=  177600             191#
UIPDR1=  177602             192#
UIPDR2=  177604             193#
UIPDR3=  177606             194#
UIPDR4=  177610             195#
UIPDR5=  177612             196#
UIPDR6=  177614             197#
```

```
UIPDR7= 177616          198#
USESTK= 000600           31#
VCIP  = 010000          416#  4598
WRITEE  005272         1778#  2297   2331   2365   2399
$BDADR  001122          481#
$BDDAT  001126          483#
$BELL   001210          509#  1074   1106
$CHARC  004030         1378*  1385   1394*  1399#
$CLR.T  024570         4745   4748#
$CMTAG  001100          469#  2051   2052   2059   2065   2066   2067
$CM1  = 000006          495#   496#   497#   498#   499#   500#   501#
$CM2  = 000014          495#   496#   497#   498#   499#   500#   501#
$CM3  = 000006          493#   495
$CM4  = 000006          501#   502#   503#   504#   505#   506#   507#
$CRLF   001215          511#  1082   1106   1164   1171   1368   1402   4740
$DBLK   004476         1505   1539   1547#
$DB20   004750         1196   1253   127(   1642#
$DOAGN  024610         4722   4743   4750   4756#
$DTBL   004466         1508   1543#
$ENDAD  024600          456   1086   2085   4752#
$ENDCT  024426         2065   4724#
$ENULL  024656         4771#
$EOP    024360         4586   4591   4711#
$EOPCT  024420         2065*  4721#  4725
$ERFLG  001103          472#   972   1012   1014   1020*  1041   1066*  1106   1901
$ERMAX  001115          478#  1014   1036*  1041   2068*
$ERROR  002534         1057#  2050
$E

RB011!4          47)#  1 56*  1077*  1078   1106   1136   5719   5723   5725   5728   5730   5738   5740
                       5745   5747   5749   5550   5752   5754   5759   5741   5763   5765   5767   5770   5774
                       5756   5578   5780   5782   5784   5787   5789   5791
$ERRTB  001334          581#  1155   1159
$ERTTL  001112          476#  1075*  1106   4737   4541*
$ESCAP  001206          508#  1035*  1092   1094   1106   2067*
$FILLC  001150          491#  1371   1402
$FILLS  001147          490#  1402
$GDADR  001120          480#
$GDDAT  001124          482#
$GET42  024552         4742#
$HD   = 000000           10
$ICNT   001104          473#  1027*  1028   1030*  1040
$ILLUP  004676         1588   1619#
$ITEMB  001114          477#  1078*  1106   1134
$LF     001216          512#  1106   1402
$LOOP   024650         4764   4768#
$LPADR  001106          474#  1018*  1033*  1038   1040   2001   2081*  2924*  3133*  3293*  3937*
$LPERR  001110          475#  1018   1034*  1040   1091   2082*  2222*  2228*  2253*  2259*  2292*  2301*  2326*
                       2335*  2360*  2369*  2394*  2403*  2439*  2451*  2460*  2489*  2503*  2515*  2544*  2561*
                       2590*  2608*  2634*  2652*  2679*  2697*  2723*  2741*  2771*  2783*  2789*  2796*  2804*
                       2837*  2864*  2879*  2925*  2930*  2951*  2970*  2985*  2992*  2999*  3006*  3034*  3056*
                       3070*  3093*  3109*  3134*  3231*  3294*  3333*  3359*  3372*  3396*  3411*  3426*  3441*
                       3456*  3471*  3486*  3501*  3516*  3531*  3546*  3557*  3584*  3621*  3662*  3678*  3717*
                       3720*  3735*  3806*  3877*  3938*  3949*  3962*  3990*  4005*  4020*  4035*  4050*  4065*
                       4080*  4095*  4110*  4125*  4140*  4151*  4178*  4215*  4256*  4272*  4314*  4317*  4332*
                       4406*  4479*
$MXCNT  002532         1031   1040#
$NULL   001146          489#  1373   1402
```

```
$NWTST= 000001          2206#  2208   2234#  2236   2274#  2276   2309#  2311   2343#  2345   2377#  2379   2419#
                        2421   2463#  2465   2521#  2523   2572#  2574   2616#  2618   2661#  2663   2705#  2707
                        2749#  2751   2807#  2809   2911#  2913   2954#  2956   3010#  3012   3074#  3076   3113#
                        3115   3209#  3211   3266#  3268   3336#  3338   3376#  3378   3560#  3562   3634#  3636
                        3690#  3692   3747#  3749   3818#  3820   3921#  3923   3966#  3968   4156#  4158   4228#
                        4230   4284#  4286   4344#  4346   4418#  4420   4491#  4493   4530#  4532   4571#  4573
$OCNT   004256          1435*  1464*  1477#
$OCTVL  005052          1644   1669#
$OMODE  004260          1430*  1434*  1439   1442*  1453*  1479#
$OVER   002516           993   1011   1019   1029   1037#
$PASS   001100           470#  1025   1041   3181   4718*  4719*  4730   4770
$PURAD  004672          1617#
$PURDN  004550          1588#  1612   2063
$PURMG  004666          1615#
$PURUP  004616          1597   1602#
$QUES   001214           510#  1106
$RDCHR= ****** U        1578
$PDDEC= ****** U        1578
$RDLIN= ****** U        1578
$RDOCT= ****** U        1578
$REGAD  001152           493#
$REG0   001154           495#  1060*  4506*  4513*  4520*  4641*  4676*  5719   5728   5738   5745   5754   5770
                        5774   5776   5778   5780   5782   5784   5787   5789   5791
$REG1   001156           496#  1061*  4507*  4514*  4521*  4642*  4677*  4678*  5730   5740   5747   5761   5763
                        5770   5774   5776   5782   5787
$REG2   001160           497#  1062*  4560*  4643*  4679*  5728   5761   5763   5772   5774   5776   5784   5787
                        5791
$REG3   001162           498#  1063*  5730   5763   5784   5789   5791
$REG4   001164           499#  1064*  5745   5784
$REG5   001166           500#  1065*
$RESRE  003562          1317#  1579
$RTRN   024646          2069   2071*  2076*  4746   4765#
$SAVR5  003524          1301#  1578
$SAVR6  004702          1596*  1602   1603*  1604*  1621#
$SCOPE  002214           981#  2057
$SETUP  000037           422#  1083   2057   2059   2061   2063   2065   2066   2067   2069   2081   2085   4716
$STUP - 177777           422#
$SVLAD  002470          1001   1032#
$SVPC   000204           454#   459
$SWR    177400             1#    10     15     16     17     18     19     20     21     22    507    508    509
                         973    974    975    976    977    992   1004   1006   1007   1012   1013   1014   1021
                        1022   1023   1034   1037   1040   1048   1049   1050   1051   1052   1053   1069   1072
                        1079   1083   1089   1106   2066   2067   2069   2081   2082   2220   2252   2290   2325
                        2359   2393   2436   2480   2542   2588   2632   2677   2721   2771   2830   2928   2969
                        3031   3092   3130   3227   3279   3351   3391   3574   3652   3708   3766   3837   3941
                        3984   4170   4246   4305   4366   4437   4497   4536   4578   4707   4717   4744   4757
                        4770
$SWRMK- 000200             1#    22     23    977    978   1008   1009
$TBIT   024654          2080*  4761*  4770#
$TIMES  001204           507#  1021*  1028   1031*  1040   2066*  2436*  2480*  2542*  2588*  2632*  2677*  2721*
                        2830*  3031*  3130*  3227*  3574*  4170*  4578*  4717*
$TKB    001140           486#
$TKS    001136           485#  2054
$TMP0   001170           501#  1784   1785*  1786   1787*  2295*  2329*  2363*  2397*  3229*  3246*  3250   3253*
                        5719   5780
$TMP1   001172           502#  4561*  5772
```

```
$TMP2   001174  503# 4562* 5772
$TMP3   001176  504#
$TMP4   001200  505# 3653* 3672 3708* 3729 3770* 3799 3841* 3870 4247* 4266 4305* 4376
                4370* 4399 4443* 4472 5767
$TMP5   001202  506# 2832* 2880 3656* 3677 3711* 3734 3769* 3805 3840* 3876 4250* 4271
                4308* 4331 4369* 4405 4442* 4478
$TN   = 000050    1#   10 2206 2217 2220# 2231 2234 2249 2252# 2262 2274 2287 2290#
                2304 2309 2322 2325# 2338 2343 2356 2359# 2372 2377 2390 2393# 2406
                2419 2433 2436# 2463 2477 2480# 2521 2539 2542# 2564 2572 2585 2588#
                2611 2616 2629 2632# 2655 2661 2674 2677# 2700 2705 2718 2721# 2744
                2749 2768 2771# 2807 2827 2830# 2911 2922 2928# 2954 2966 2969# 3010
                3028 3031# 3074 3089 3092# 3113 3127 3130# 3131 3182 3209 3224 3227#
                3260 3266 3276 3279# 3297 3336 3348 3351# 3376 3388 3391# 3560 3571
                3574# 3634 3649 3652# 3679 3690 3705 3708# 3736 3747 3763 3766# 3768
                3807 3818 3834 3837# 3921 3935 3941# 3966 3981 3984# 4156 4167 4170#
                4228 4243 4246# 4273 4284 4302 4305# 4333 4344 4363 4366# 4368 4407
                4418 4437# 4441 4480 4491 4497# 4530 4536# 4571 4578#
$TPB    001144  488# 1391* 1402
$TPFLG  001151  492# 1351 1402
$TPS    001142  487# 1389 1402
$TRAP   004506 1557# 2061
$TRP  = 000022 1564# 1574# 1575# 1576# 1577# 1578# 1579# 1580# 1581# 1582#
$TRPAD  004526 1561 1572#
$TSTNM  001102  471#  972 1010 1032* 1037 1041 1058 1068 1106 2926* 2927 3135* 3136
                3939* 3940 4716*
$TYPBN= ****** U 1578
$TYPDS  004262 1493# 1577
$TYPE   003620 1351# 1564 1573
$TYPEC  003764 1370 1377 1384 1389# 1390
$TYPEX  004032 1395 1397 1400#
$TYPOC  004060 1433# 1574
$TYPON  004074 1432 1435# 1576
$TYPOS  004034 1428# 1575
$XTSTR  002266  995#
$$GET4= 000001 4744# 4749
$$TRP = 000002 1563# 1574 1575 1576 1577 1578 1579 1580 1581 1582
$OFILL  004257 1429* 1433* 1443 1478#
.     - 042060  425#  429  431#  454  455#  457#  459#  467#  513 1040 1041 1106 1223#
                1402 1547# 1599 1620 1669# 2048# 2055 2081 2082 2090# 4555 4729# 4770
                4772# 5854# 5856 5859# 5861#
```

AND	1#	1755	1781	1785	1808	1812	1834	1838	1842	1865	1869	1894	1898	3315	3319
COMMEN	1#	414#													
COUNT1	1#	2589	2633	2678	2722										
DONE	1#	4712													
ENDCOM	1#	414#													
ENTRY	1#	2922	3131	3935											
ERROR	32#	1752	1760	1762	1790	1792	1817	1819	1847	1849	1874	1876	1903	1905	1943
	1945	1988	1997	2036	2232	2263	2305	2339	2373	2407	2445	2459	2497	2514	2565
	2612	2656	2701	2745	2782	2788	2795	2803	2849	2878	2944	2981	2991	2998	3005
	3044	3068	3108	3155	3177	3187	3252	3261	3326	3328	3332	3371	3406	3421	3436
	3451	3466	3481	3496	3511	3526	3541	3556	3605	3629	3674	3731	3802	3873	3961
	4000	4015	4030	4045	4060	4075	4090	4105	4120	4135	4150	4199	4223	4268	4328
	4402	4475	4508	4515	4522	4563	4644	4680							
ESCAPE	1#	414#													
MAPADD	1#	3396	3411	3426	3441	3456	3471	3486	3501	3516	3531	3546	3990	4005	4020
	4035	4050	4065	4080	4095	4110	4125	4140							
MSG1	2206#	2208													
MSG10	2463#	2465													
MSG11	2521#	2523													
MSG12	2572#	2574													
MSG13	2616#	2618													
MSG14	2661#	2663													
MSG15	2705#	2707													
MSG16	2749#	2751													
MSG17	2807#	2809													
MSG2	2234#	2236													
MSG20	2911#	2913													
MSG21	2954#	2956													
MSG22	3010#	3012													
MSG23	3074#	3076													
MSG24	3113#	3115													
MSG25	3209#	3211													
MSG26	3266#	3268													
MSG27	3336#	3338													
MSG3	2274#	2276													
MSG30	3376#	3378													
MSG31	3560#	3562													
MSG32	3634#	3636													
MSG33	3690#	3692													
MSG34	3747#	3749													
MSG35	3818#	3820													
MSG36	3921#	3923													
MSG37	3966#	3968													
MSG4	2309#	2311													
MSG40	4155#	4158													
MSG41	4228#	4230													
MSG42	4284#	4286													
MSG43	4344#	4346													
MSG44	4418#	4420													
MSG45	4490#	4493													
MSG46	4529#	4532													
MSG47	4571#	4573													
MSG5	2343#	2345													
MSG6	2377#	2379													
MSG7	2419#	2421													
MULT	1#	414#													

```
NEWTST    1#  414#  2206  2234  2274  2309  2343  2377  2419  2463  2521  2572  2616  2661  2705
         2749  2807  2911  2954  3010  3074  3113  3209  3266  3336  3376  3560  3634  3690  3747
         3818  3921  3966  4156  4228  4284  4344  4418  4491  4530  4571
POP       1#  414#  1322  1534  1606
PUSH      1#  414#  1302  1493  1590
SAVTST    1#  1058
SCOPE    33#  2217  2249  2287  2322  2356  2390  2433  2477  2539  2585  2629  2674  2718  2768
         2827  2881  2923  2966  3028  3089  3127  3132  3224  3276  3348  3388  3571  3649  3705
         3763  3834  3895  3936  3981  4167  4243  4302  4363  4436  4496  4535  4577  4712
SETTRA  1564#  1574  1575  1576  1577  1578  1579  1580  1581
SETUP     1#  414#  2050
SKIP      1#  414#  2231  2262  2304  2338  2372  2406  2564  2611  2655  2700  2744  3182  3260
         3297  3679  3736  3768  3807  3839  3878  4273  4333  4368  4407  4441  4480
SLASH     1#  414#
SPACE   414#
SSCOPE    1#  982  2850
STARS     1#  414#  434  460  565  966  1041  1283  1329  1402  1480  1548  1583  1630  2206
         2215  2234  2247  2274  2285  2309  2320  2343  2354  2377  2388  2419  2431  2463  2475
         2521  2537  2572  2583  2616  2627  2661  2672  2705  2716  2749  2766  2807  2825  2911
         2921  2954  2964  3010  3026  3074  3087  3113  3125  3209  3222  3266  3274  3336  3346
         3376  3386  3560  3569  3634  3647  3690  3703  3747  3761  3818  3832  3921  3934  3966
         3979  4156  4165  4228  4241  4284  4300  4344  4361  4412  4435  4491  4495  4530  4534
         4571  4576  4699
TRACK     1#  2216  2248  2286  2321  2355  2389  2432  2476  2538  2584  2628  2673  2717  2767
         2826  2965  3027  3088  3126  3223  3275  3347  3387  3570  3648  3704  3762  3833  3980
         4166  4242  4301  4362
TRACK1    1#  2217  2249  2287  2322  2356  2390  2433  2477  2539  2585  2629  2674  2718  2768
         2827  2966  3028  3089  3127  3224  3276  3348  3388  3571  3649  3705  3763  3834  3981
         4167  4243  4302  4363
TRMTRP  1564#
TYPBIN    1#  414#
TYPDEC    1#  414#  4730  4737
TYPNAM    1#  414#  2083
TYPNUM    1#  414#
TYPOCS    1#  414#
TYPOCT    1#  414#
TYPTXT    1#  414#  4726  4733
USER      1#  513
ZEROER    1#  1096
$$CMRE  460#  495  496  497  498  499  500
$$CMTM  460#  501  502  503  504  505  506
$$ESCA    1#  414#
$$NEWT    1#  414#  2206  2234  2274  2309  2343  2377  2419  2463  2521  2572  2616  2661  2705
         2749  2807  2911  2954  3010  3074  3113  3209  3266  3336  3376  3560  3634  3690  3747
         3818  3921  3966  4156  4228  4284  4344  4418  4491  4530  4571
$$SET   1564#  1574  1575  1576  1577  1578  1579  1580  1581
$$SETU  2069#
$$SKIP    1#  414#  2231  2262  2304  2338  2372  2406  2564  2611  2655  2700  2744  3182  3260
         3297  3679  3736  3768  3807  4273  4333  4368  4407  4441  4480
.EQUAT    1#
.HEADE    1#
.KT11     1#
.SETUP    1#  422
.SWRHI    1#  10
.SWRLO   23#
.$ACT1    1#  434
```

```
.$CATC      1#    422
.$CMTA      1#    460
.$DB2D      1#
.$DB2O      1#   1630
.$DIV       1#
.$EOP       1#   4699
.$ERRO      1#   1041
.$ERRT      1#
.$MULT      1#
.$POWE      1#   1583
.$RAND      1#
.$RDDE      1#
.$RDOC      1#
.$READ      1#
.$SAVE      1#   1283
.$SB2D      1#
.$SB2O      1#
.$SCOP      1#    966
.$SIZE      1#
.$SUPR      1#
.$TRAP      1#   1548
.$TYPB      1#
.$TYPD      1#   1480
.$TYPE      1#   1329
.$TYPO      1#   1402
.1170       1#     24


. ABS.  042060     000

ERRORS DETECTED:  0

DSKZ:CEKBFD.BIN,DSKZ:CEKBFD.LST/CRF/SOL/NL:TOC=DSKZ:CEKBFD.SML,CEKBFD.P11
RUN-TIME:  48 68  5 SECONDS
RUN-TIME RATIO: 237/123=1.9
CORE USED:  34K  (67 PAGES)
```