

11/70-11/74

11/70 UNIBUS MAP
CEKBFDO

AH-7980D-MC
FICHE 1 OF 1

NOV 1980
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

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

DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL.

COPYRIGHT (C) 1975,1980 BY DIGITAL EQUIPMENT CORPORATION

THE FOLLOWING ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION:

DIGITAL
DEC

PDP
DECUS

UNIBUS
DECTAPE

MASSBUS
DECK/11

TABLE OF CONTENTS

1) ABSTRACT

2) REQUIREMENTS

2.1 EQUIPMENT

2.2 STORAGE

2.3 PRELIMINARY PROGRAMS

3) LOADING PROCEDURE

3.1 METHOD

4) STARTING PROCEDURE

4.1 STARTING ADDRESS

4.2 PROGRAM AND OPERATOR ACTION

4.3 SPECIAL STARTING PROCEDURE

5) OPERATING PROCEDURE

5.1 OPERATIONAL SWITCH SETTINGS

5.2 SUB-ROUTINE ABSTRACTS

5.3 PROGRAM AND OPERATOR ACTION

6) ERRORS

6.1 ERROR HALTS AND DESCRIPTION

6.2 ERROR RECOVERY

6.3 SAMPLE ERROR MESSAGES

7) RESTRICTIONS

7.1 STARTING RESTRICTIONS

7.2 OPERATING RESTRICTIONS

8) MISCELLANEOUS

8.1 EXECUTION TIME

8.2 ADDRESS GENERATION IN THE PDP-11/70

9) REVISION HISTORY

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.

2.2

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 WILL LOAD AND RUN IN.

3. LOADING PROCEDURE

3.1 METHOD.

THIS PROGRAM CAN BE LOADED FROM ANY DEVICE THAT IS SUPPORTED BY XXDP AND SHOULD BE LOADED USING THE XXDP PROCEDURE FOR THAT DEVICE.

4. STARTING PROCEDURE

4.1 STARTING ADDRESS

PROGRAM STARTS AT ADDRESS 200

4.2 PROGRAM AND/OR OPERATOR ACTION

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

4.3 SPECIAL STARTING PROCEDURE

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

DEFINITION OF THE BITS IN THE CACHE CONTROL REGISTER:

BIT00 -DISABLE TRAPS
BIT01 -DISABLE UNIBUS TRAPS
BIT02 -FORCE MISS ON READ, GROUP 0
BIT03 -FORCE MISS ON READ, GROUP 1
BIT04 -FORCE REPLACEMENT IN GROUP 0
BIT05 -FORCE REPLACEMENT IN GROUP 1

5. OPERATING PROCEDURE

5.1 OPERATIONAL SWITCH SETTINGS

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

5.2 SUB-ROUTINE ABSTRACTS

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

5.2.1 MACRO LIBRARY SUBROUTINES (FOUND IN MOST PROGRAMS)

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

5.2.2 SUBROUTINES UNIQUE TO THIS PROGRAM

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

5.2.3 TRAP AND ABORT HANDLER ROUTINES

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

6. ERRORS

6.1 ERROR HALTS AND DESCRIPTION

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

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

6.2 ERROR RECOVERY

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

6.3 SAMPLE ERROR TYPE OUTS
SEE '\$ERRTB:' FOR SAMPLE ERROR TYPEOUTS.

6.3.1 MULTIPLE TYPE ERRORS

THE FOLLOWING REGISTERS TIMED OUT WHEN REFERENCED

REG.ADR	TESTNO	ERRORPC
170210	000001	015226
170212	000001	015232
170214	000001	015232
170216	000001	015232
170230	000001	015232
170232	000001	015232
170234	000001	015232
170236	000001	015232
170250	000001	015232
170252	000001	015232
170254	000001	015232
170256	000001	015232
170270	000001	015232
170272	000001	015232
170274	000001	015232
170276	000001	015232
170310	000001	015232
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		TESTNO	ERRORPC
'OR'	'AND'	#ERRORS		
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 D0 TESTS 27 AND 36 WILL BE BYPASSED ON SYSTEMS WITH
GREATER THAN 1920K OF MEMORY.

2
.END

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56

```
.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= R0           ;;GENERAL REGISTER
000001 R1= R1           ;;GENERAL REGISTER
000002 R2= R2           ;;GENERAL REGISTER
000003 R3= R3           ;;GENERAL REGISTER
000004 R4= R4           ;;GENERAL REGISTER
000005 R5= R5           ;;GENERAL REGISTER
000006 R6= R6           ;;GENERAL REGISTER
000007 R7= R7           ;;GENERAL REGISTER
.EQUIV R0,R10          ;;GENERAL REGISTER
```

```
57 .EQUIV R1,R11          ;;GENERAL REGISTER
58 .EQUIV R2,R12          ;;GENERAL REGISTER
59 .EQUIV R3,R13          ;;GENERAL REGISTER
60 .EQUIV R4,R14          ;;GENERAL REGISTER
61 .EQUIV R5,R15          ;;GENERAL REGISTER
62 000006 SP=%6          ;;STACK POINTER
63 .EQUIV SP,KSP          ;;KERNEL STACK POINTER
64 .EQUIV SP,SSP          ;;SUPERVISOR STACK POINTER
65 .EQUIV SP,USP          ;;USER STACK POINTER
66 000007 PC=%7          ;;PROGRAM COUNTER
67
68 ;*PRIORITY LEVEL DEFINITIONS
69 000000 PR0= 0          ;;PRIORITY LEVEL 0
70 000040 PR1= 40         ;;PRIORITY LEVEL 1
71 000100 PR2= 100        ;;PRIORITY LEVEL 2
72 000140 PR3= 140        ;;PRIORITY LEVEL 3
73 000200 PR4= 200        ;;PRIORITY LEVEL 4
74 000240 PR5= 240        ;;PRIORITY LEVEL 5
75 000300 PR6= 300        ;;PRIORITY LEVEL 6
76 000340 PR7= 340        ;;PRIORITY LEVEL 7
77
78 ;*'SWITCH REGISTER' SWITCH DEFINITIONS
79 100000 SW15= 100000
80 040000 SW14= 40000
81 020000 SW13= 20000
82 010000 SW12= 10000
83 004000 SW11= 4000
84 002000 SW10= 2000
85 001000 SW09= 1000
86 000400 SW08= 400
87 000200 SW07= 200
88 000100 SW06= 100
89 000040 SW05= 40
90 000020 SW04= 20
91 000010 SW03= 10
92 000004 SW02= 4
93 000002 SW01= 2
94 000001 SW00= 1
95 .EQUIV SW09,SW9
96 .EQUIV SW08,SW8
97 .EQUIV SW07,SW7
98 .EQUIV SW06,SW6
99 .EQUIV SW05,SW5
100 .EQUIV SW04,SW4
101 .EQUIV SW03,SW3
102 .EQUIV SW02,SW2
103 .EQUIV SW01,SW1
104 .EQUIV SW00,SW0
105
106 ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
107 100000 BIT15= 100000
108 040000 BIT14= 40000
109 020000 BIT13= 20000
110 010000 BIT12= 10000
111 004000 BIT11= 4000
112 002000 BIT10= 2000
```

```

13 001000 BIT09= 1000
14 000400 BIT08= 400
15 000200 BIT07= 200
16 000100 BIT06= 100
17 000040 BIT05= 40
18 000020 BIT04= 20
19 000010 BIT03= 10
20 000004 BIT02= 4
21 000002 BIT01= 2
22 000001 BIT00= 1
23 .EQUIV BIT09,BIT9
24 .EQUIV BIT08,BIT8
25 .EQUIV BIT07,BIT7
26 .EQUIV BIT06,BIT6
27 .EQUIV BIT05,BIT5
28 .EQUIV BIT04,BIT4
29 .EQUIV BIT03,BIT3
30 .EQUIV BIT02,BIT2
31 .EQUIV BIT01,BIT1
32 .EQUIV BIT00,BIT0

```

```

33 ;*BASIC 'CPU' TRAP VECTOR ADDRESSES
34 ERRVEC= 4 ;:TIME OUT AND OTHER ERRORS
35 RESVEC= 10 ;:RESERVED AND ILLEGAL INSTRUCTIONS
36 TBITVEC=14 ;: 'T' BIT
37 TRTVEC= 14 ;:TRACE TRAP
38 BPTVEC= 14 ;:BREAKPOINT TRAP (BPT)
39 IOTVEC= 20 ;:INPUT/OUTPUT TRAP (IOT) **SCOPE**
40 PWRVEC= 24 ;:POWER FAIL
41 EMTVEC= 30 ;:EMULATOR TRAP (EMT) **ERROR**
42 TRAPVEC=34 ;: 'TRAP' TRAP
43 TKVEC= 60 ;:TTY KEYBOARD VECTOR
44 TPVEC= 64 ;:TTY PRINTER VECTOR
45 CACHVEC=114 ;:CACHE ERROR INTERRUPT VECTOR
46 PIRQVEC=240 ;:PROGRAM INTERRUPT REQUEST VECTOR
47 MMVEC= 250 ;:MEMORY MANAGEMENT VECTOR

```

149 .SBTTL CACHE REGISTER DEFINITIONS

```

150
151
152
153 LOADRS = 177740 ;:LOWER 16 BITS OF ADDRESS THAT CAUSED ERROR
154 HIADRS = 177742 ;:UPPER SIX BITS OF ADDRESS THAT CAUSED ERROR
155 MEMERR = 177744 ;:CACHE ERROR REGISTER
156 CONTRL = 177746 ;:MEMORY CONTROL REGISTER
157 MAINT = 177750 ;:MEMORY MAINTENANCE REGISTER
158 HITMIS = 177752 ;:HIT MISS REGISTER '1' IMPLIES HIT IN CACHE
159

```

160 .SBTTL CPU REGISTER DEFINITIONS

```

161
162
163
164 SIZELO = 177760 ;:MEMORY SIZE REGISTER NUMBER TO PUT INTO A PAR
165 SIZEHI = 177762 ;:TO GET TO THE LAST 32 WORDS OF MEMORY
166 SYSTID = 177764 ;:HIGH SIZE REGISTER, RESERVED FOR FUTURE USE
167 ;:CURRENTLY ALL ZERO
168 ;: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
177574
177576
172516

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

;*USER 'I' PAGE DESCRIPTOR REGISTERS

177600
177602
177604
177606
177610
177612
177614
177616

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

;*USER 'D' PAGE DESCRIPTOR REGISTORS

177620
177622
177624
177626
177630
177632
177634
177636

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

;*USER 'I' PAGE ADDRESS REGISTERS

177640
177642
177644
177646
177650
177652
177654
177656

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

;*USER 'D' PAGE ADDRESS REGISTERS

177660

UDPAR0= 177660

225	177662	UDPAR1= 177662
226	177664	UDPAR2= 177664
227	177666	UDPAR3= 177666
228	177670	UDPAR4= 177670
229	177672	UDPAR5= 177672
230	177674	UDPAR6= 177674
231	177676	UDPAR7= 177676

;*SUPERVISOR 'I' PAGE DESCRIPTOR REGISTERS

232		
233		
234		
235	172200	SIPDR0= 172200
236	172202	SIPDR1= 172202
237	172204	SIPDR2= 172204
238	172206	SIPDR3= 172206
239	172210	SIPDR4= 172210
240	172212	SIPDR5= 172212
241	172214	SIPDR6= 172214
242	172216	SIPDR7= 172216

;*SUPERVISOR 'D' PAGE DESCRIPTOR REGISTERS

243		
244		
245		
246	172220	SDPDR0= 172220
247	172222	SDPDR1= 172222
248	172224	SDPDR2= 172224
249	172226	SDPDR3= 172226
250	172230	SDPDR4= 172230
251	172232	SDPDR5= 172232
252	172234	SDPDR6= 172234
253	172236	SDPDR7= 172236

;*SUPERVISOR 'I' PAGE ADDRESS REGISTERS

254		
255		
256		
257	172240	SIPAR0= 172240
258	172242	SIPAR1= 172242
259	172244	SIPAR2= 172244
260	172246	SIPAR3= 172246
261	172250	SIPAR4= 172250
262	172252	SIPAR5= 172252
263	172254	SIPAR6= 172254
264	172256	SIPAR7= 172256

;*SUPERVISOR 'D' PAGE ADDRESS REGISTERS

265		
266		
267		
268	172260	SDPAR0= 172260
269	172262	SDPAR1= 172262
270	172264	SDPAR2= 172264
271	172266	SDPAR3= 172266
272	172270	SDPAR4= 172270
273	172272	SDPAR5= 172272
274	172274	SDPAR6= 172274
275	172276	SDPAR7= 172276

;*KERNEL 'I' PAGE DESCRIPTOR REGISTERS

276		
277		
278		
279	172300	KIPDR0= 172300
280	172302	KIPDR1= 172302

281 172304
282 172306
283 172310
284 172312
285 172314
286 172316
287
288
289
290 172320
291 172322
292 172324
293 172326
294 172330
295 172332
296 172334
297 172336
298
299
300
301 172340
302 172342
303 172344
304 172346
305 172350
306 172352
307 172354
308 172356
309
310
311
312 172360
313 172362
314 172364
315 172366
316 172370
317 172372
318 172374
319 172376
320
321
322
323
324
325
326
327
328
329
330
331 170200
332 170202
333 170204
334 170206
335 170210
336 170212

KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316

.*KERNEL 'D' PAGE DESCRIPTOR REGISTERS

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

.*KERNEL 'I' PAGE ADDRESS REGISTERS

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

.*KERNEL 'D' PAGE ADDRESS REGISTERS

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

.SRTL 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'

MAPL00 = 170200
MAPH00 = 170202
MAPL01 = 170204
MAPH01 = 170206
MAPL02 = 170210
MAPH02 = 170212

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

393 170374
394 170376
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415 100000
416 010000
417 000040
418 000020
419 000054
420 000034
421 177744
422
423
424
425 000000
426
427
428
429
430
431 000200
432
433 000200 000137 010000
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448

MAPL37 = 170374
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

BYP=BIT15
VCIP=BIT12
S1=BIT5
S0=BIT4
SIMOM1=BIT5+BIT3+BIT2
SOMOM1=BIT4+BIT3+BIT2
MSER=177744

.SBTTL TRAP CATCHER

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

.SBTTL STARTING ADDRESS(ES)
. =200

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 LOGICAL
:*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
:*

449
450
451
452
453
454 000204
455 000046
456 000046 024600
457 000052
458 000052 000000
459 000204

```

: * 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
.=46            ;;SET LOCATION COUNTER
.WORD SENDAD    ;;SET LOC.46 TO ADDRESS SENDAD
.=52            ;;SET LOCATION COUNTER
.WORD 0         ;;SET LOC.52 TO ZERO
.= $SVPC        ;;RESTORE LOCATION COUNTER

```

460
461
462
463
464
465
466
467 001100
468
469 001100
470 001100 000000
471 001102 000
472 001103 000
473 001104 000000
474 001106 000000
475 001110 000000
476 001112 000000
477 001114 000
478 001115 001
479 001116 000000
480 001120 000000
481 001122 000000
482 001124 000000
483 001126 000000
484 001130 000000 000000 000000
485 001136 177560
486 001140 177562
487 001142 177564
488 001144 177566
489 001146 000
490 001147 002
491 001150 012
492 001151 000
493 001152 000000
494
495 001154 000000
496 001156 000000
497 001160 000000
498 001162 000000
499 001164 000000
500 001166 000000
501 001170 000000
502 001172 000000
503 001174 000000
504 001176 000000
505 001200 000000
506 001202 000000
507 001204 000000
508 001206 000000
509 001210 177607 000377
510 001214 077
511 001215 015
512 001216 000012
513 001220 000000
514
515 001222 000000

::*****

.SBTTL COMMON TAGS

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

.-1100

\$CMTAG: .WORD 0
\$PASS: .WORD 0
\$STNM: .BYTE 0
\$ERFLG: .BYTE 0
\$ICNT: .WORD 0
\$LPADR: .WORD 0
\$LPERR: .WORD 0
\$ERTTL: .WORD 0
\$ITEMB: .BYTE 0
\$ERMAX: .BYTE 1
\$ERRPC: .WORD 0
\$GDADR: .WORD 0
\$BDADR: .WORD 0
\$GDDAT: .WORD 0
\$BDDAT: .WORD 0
\$TKS: 177560
\$TKB: 177562
\$TPS: 177564
\$TPB: 177566
\$NULL: .BYTE 0
\$FILLS: .BYTE 2
\$FILLC: .BYTE 12
\$TPFLG: .BYTE 0
\$REGAD: .WORD 0
\$REG0: .WORD 0
\$REG1: .WORD 0
\$REG2: .WORD 0
\$REG3: .WORD 0
\$REG4: .WORD 0
\$REG5: .WORD 0
\$TMP0: .WORD 0
\$TMP1: .WORD 0
\$TMP2: .WORD 0
\$TMP3: .WORD 0
\$TMP4: .WORD 0
\$TMP5: .WORD 0
\$TIMES: 0
\$ESCAPE: 0
\$BELL: .ASCII <207><377><377>
\$QUES: .ASCII /?/
\$CRLF: .ASCII <15>
\$LF: .ASCII <12>
PADRS: .WORD 0
PADRSH: .WORD 0

:: START OF COMMON TAGS
:: CONTAINS PASS COUNT
:: CONTAINS THE TEST NUMBER
:: CONTAINS ERROR FLAG
:: CONTAINS SUBTEST ITERATION COUNT
:: CONTAINS SCOPE LOOP
:: CONTAINS SCOPE RETURN FOR ERRORS
:: CONTAINS TOTAL ERRORS DETECTED
:: CONTAINS ITEM CONTROL BYTE
:: CONTAINS MAX. ERRORS PER TEST
:: CONTAINS PC OF LAST ERROR INSTRUCTION
:: CONTAINS OF 'GOOD' DATA
:: CONTAINS OF 'BAD' DATA
:: CONTAINS 'GOOD' DATA
:: CONTAINS 'BAD' DATA
:: RESERVED—NOT TO BE USED
:: TTY KBD STATUS
:: TTY KBD BUFFER
:: TTY PRINTER STATUS REG.
:: TTY PRINTER BUFFER REG.
:: CONTAINS NULL CHARACTER FOR FILLS
:: CONTAINS # OF FILLER CHARACTERS REQUIRED
:: INSERT FILL CHARS. AFTER A 'LINE FEED'
:: 'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)
:: CONTAINS THE FROM
:: WHICH (\$REG0) WAS OBTAINED
:: CONTAINS ((REGAD)+0)
:: CONTAINS ((REGAD)+2)
:: CONTAINS ((REGAD)+4)
:: CONTAINS ((REGAD)+6)
:: CONTAINS ((REGAD)+10)
:: CONTAINS ((REGAD)+12)
:: USER DEFINED
:: USER DEFINED
:: USER DEFINED
:: USER DEFINED
:: USER DEFINED
:: USER DEFINED
:: USER DEFINED
:: MAX. NUMBER OF ITERATIONS
:: ESCAPE ON ERROR
:: CODE FOR BELL
:: QUESTION MARK
:: CARRIAGE RETURN
:: LINE FEED
:: HOLDS THE LOWER 16 BITS OF A 22 BIT
:: ADDRESS GENERATED FOR TYPE OUT.
:: HOLDS THE UPPER 6 BITS OF A 22 BIT

516					.ADDRESS GENERATED FOR TYPE OUT
517	001224	000000	ADRAND: .WORD	0	:LOGICAL AND OF FAILING ADDRESSES
518	001226	000000	ADDRCR: .WORD	0	:LOGICAL OR OF FAILING ADDRESSES
519	001230	000000	DATAND: .WORD	0	:LOGICAL AND OF BAD DATA
520	001232	000000	DATAOR: .WORD	0	:LOGICAL OR OF BAD DATA
521	001234	000000	PATAND: .WORD	0	:LOGICAL AND OF PATTERN LOADED
522	001236	000000	PATOR: .WORD	0	:LOGICAL OR OF PATTERN LOADED
523	001240	000000	LOWEST: .WORD	0	:HOLDS NUMBER TO PUT IN PAR TO CAUSE THE :LOWEST USEABLE MAP REGISTER TO RESPOND
524					:HOLDS NUMBER TO PUT IN PAR TO CAUSE THE :HIGHEST USEABLE MAP REGISTER TO RESPOND
525	001242	000000	HIGEST: .WORD	0	:HIGHEST USEABLE MAP REGISTER TO RESPOND
526					:HOLDS I/O PAGE ADDR OF LOW 16 BITS OF :THE LOWEST USEABLE MAP REGISTER
527	001244	000000	LREGL: .WORD	0	:HOLDS I/O PAGE ADDR OF HIGH 16 BITS OF :OF THE LOWEST USEABLE MAP REGISTER
528					:HOLDS I/O PAGE ADDR OF LOW 16 BITS OF :THE HIGHEST USEABLE MAP REGISTER
529	001246	000000	LREGU: .WORD	0	:HOLDS I/O PAGE ADDR OF HIGH 16 BITS OF :THE HIGHEST USEABLE MAP REGISTER
530					:MULTIPLE ERROR ERROR COUNTER
531	001250	000000	HREGL: .WORD	0	:AUXILIARY COUNTER
532					:FLAG TO INDICATE TO LAST PROGRAM PASS N
533	001252	000000	HREGU: .WORD	0	:HOLDS TEST NUMBER FOR ERROR TYPE OUTS
534					:HOLDS THE EXPECTED CPU ERROR CODE
535	001254	000000	ERRCNT: .WORD	0	:HOLDS RECEIVED CPU ERROR CONDITION
536	001256	000000	CNTR: .WORD	0	:HOLDS LOWER 16 BITS OF CACHE ADDR
537	001260	000000	FLAG: .WORD	0	:HOLDS UPPER 6 BITS OF CACHE ADDR
538	001262	000000	TESTNO: .WORD	0	:HOLDS RECEIVED PARITY ERROR CONDITION
539	001264	000000	CPUEXP: .WORD	0	:HOLDS CONTENTS OF CONTROL REGISTER
540	001266	000000	PCPUER: .WORD	0	:HOLDS CONTENTS OF MAINTENANCE REGISTER
541	001270	000000	PLOADR: .WORD	0	:HOLDS PC OF INST THAT CAUSED TRAP
542	001272	000000	PHIADR: .WORD	0	:HOLDS THE RETURN ADDRESS AFTER A TRAP
543	001274	000000	PPARER: .WORD	0	:HOLDS THE OLD PROCESSOR STATUS
544	001276	000000	PCONTR: .WORD	0	:HOLDS OLD PSW FOR TBITSTORE
545	001300	000000	PMAINT: .WORD	0	:HOLDS CONTENTS OF PPR0 AFTER TRAP
546	001302	000000	BADPC: .WORD	0	:HOLDS CONTENTS OF PPR1 AFTER TRAP
547	001304	000000	OLDPC: .WORD	0	:HOLDS CONTENTS OF PPR2 AFTER TRAP
548	001306	000000	CLDPS: .WORD	0	:WILL HOLD P.A.R. DATA FOR TOP OF MEMORY
549	001310	000000	OLDPSW: .WORD	0	:RETRY FLAG IN CASE OF PARITY ABORTS
550	001312	000000	PPR0: .WORD	0	:LOCATION TO HOLD ESCAPE ADDRESS ON :PARITY ERRORS.
551	001314	000000	PPR1: .WORD	0	:PATTERN TO BE USED TO LOAD INTO MEMORY
552	001316	000000	PPR2: .WORD	0	:1174 WITHOUT MP CACHE FLAG
553	001320	000000	RSIZE: .WORD	0	:1174 WITH MP CACHE FLAG
554	001322	000000	RETRY: .WORD	0	:KB11CM FLAG (1170 WITH MP MODS)
555	001324	000000	NXTTST: .WORD	0	:CISP OPTION PRESENT FLAG
556					
557	001326	000200	DATA: .WORD	200	
558	001330	000	KB11E: .BYTE	0	
559	001331	000	KB11EM: .BYTE	0	
560	001332	000	KB11CM: .BYTE	0	
561	001333	000	CISP: .BYTE	0	
562					
563					
564		000007			:OPCODE FOR MFPT INSTRUCTION (AVAILABLE ON KB11-E AND KB11-EM ONLY) MFPT=7

565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620

::*****

.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

\$ERRTB:

:ITEM1

EM1 :NOT THE CORRECT CPU TRAP CONDITION THRU ERRVEC (#004)
DH1 :RECEIVD EXPECTD TESTNO PC AT ABORT
DT1 :PCPUER,CPUEXP,TESTNO,BADPC,0
DF1 : 0. 0. 0. 0

:ITEM 2

EM2 :UNEXPECTED CPU TRAP THRU ERRVEC (#004)
DH2 :RECEIVD TESTNO PC AT ABORT
DT2 :PCPUER,TESTNO,BADPC
DF2 : 0. 0. 0

:ITEM 3

EM3 :UNEXPECTED CACHE PARITY ERROR THRU CACHVEC (#114)
DH3 :WILL RETRY TEST ONCE
DT3 :PARITY ADDRESS MAINTEN CONTROL
DF3 :CONDITN REFERENC'D REGISTR REGISTR TESTNO PC AT ABORT
PPARER,LOADDR,PMaint,PCONTR,TESTNO,BADPC,0
: 0. 2. 0. 0. 0. 0

:ITEM 4

EM4 :UNEXPECTED MAIN MEMORY PARITY ERROR THRU CACHVEC (#114)
DH3 :WILL RETRY TEST ONCE
DT3 :PARITY ADDRESS MAINTEN CONTROL
DF3 :CONDITN REFERENC'D REGISTR REGISTR TESTNO PC AT ABORT
PPARER,LOADDR,PMaint,PCONTR,TESTNO,BADPC,0
: 0. 2. 0. 0. 0. 0

:ITEM 5

EM5 :MEMORY MANAGEMENT TRAP, MEMORY MANAGEMENT STATUS REGISTERS
DH5 :STATUS AUTOI/D VIRTADR
DT5 :REGISTR REGISTR REGISTR TESTNO PC AT ABORT
DF5 :PMR0,PMR1,PMR2,TESTNO,BADPC,0
: 0. 0. 0. 0. 0

:ITEM 6

EM6 :SUMMARY OF MAP REGISTERS THAT TIMED LIT ON READ
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
627				: 'OR' 'AND' #ERRORS TESTNO ERRORPC
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
634				: 'OR' 'AND' 'OR' 'AND' #ERRORS TESTNO ERRORPC
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
641				: 'OR' 'AND' 'OR' 'AND' #ERRORS TESTNO ERRORPC
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
648				:EXPECTED RECEIVD TESTNO ERRORPC
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
655				:EXPECTED RECEIVD TESTNO ERRORPC
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
662				: 'OR' 'AND' 'OR' 'AND' #ERRORS TESTNO
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
669				: 'OR' 'AND' 'OR' 'AND' 'OR' 'AND' #ERRORS TESTNO
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
676				: 'OR' 'AND' 'OR' 'AND' 'OR' 'AND' #ERRORS TESTNO

677	001510	037150	DI15	:ADDROR,ADRAND,PATTOR,PATAND,DATAOR,DATAND,ERRCNT,TESTNO,0
678	001512	037656	DF15	: 0, 0, 0, 0, 0, 0, 1, 0
679				
680			:ITEM 17	
681	001514	026463	EM17	:COULD NOT CLEAR CACHE CONTROL REGISTER
682				:POSSIBLE ERROR IN CACHE REGISTER DATA PATH
683	001516	034757	DH17	:RECEIVED TESTNO ERRORPC
684	001520	037172	DT17	:SREG0,TESTNO,SERRPC,0
685	001522	037666	DF17	: 0, 0, 0
686				
687			:ITEM 20	
688	001524	026605	EM20	:COULD NOT CLEAR CACHE MAINTENANCE REGISTER
689				:POSSIBLE ERROR IN CACHE REGISTER DATA PATH
690	001526	034757	DH17	:RECEIVED TESTNO ERRORPC
691	001530	037202	DT20	:SREG1,TESTNO,SERRPC,0
692	001532	037666	DF17	: 0, 0, 0
693				
694			:ITEM 21	
695	001534	026733	EM21	:COULD NOT READ 177740 FROM CACHE LO ADDRESS REG (LOADRS)
696				:POSSIBLE ERROR IN CACHE REGISTER DATA PATH
697	001536	034757	DH17	:RECEIVED TESTNO ERRORPC
698	001540	037172	DT17	:SREG0,TESTNO,SERRPC,0
699	001542	037666	DF17	: 0, 0, 0
700				
701			:ITEM 22	
702	001544	027077	EM22	:COULD NOT READ 000003 FROM CACHE HI ADDRESS REG (HIADRS)
703				:POSSIBLE ERROR IN CACHE REGISTER DATA PATH
704	001546	034757	DH17	:RECEIVED TESTNO ERRORPC
705	001550	037172	DT17	:SREG0,TESTNO,SERRPC,0
706	001552	037666	DF17	: 0, 0, 0
707				
708			:ITEM 23	
709	001554	027243	EM23	:SUMMARY OF COUNT PATTERN FAILURES IN CACHE CONTROL REGISTER
710	001556	035007	DH23	:EXPECTD EXPECTD RECEIVED RECEIVED
711				:OR AND OR AND #ERRORS TESTNO
712	001560	037212	DT23	:PATTOR,PATAND,DATAOR,DATAND,ERRCNT,TESTNO,0
713	001562	037671	DF23	: 0, 0, 0, 0, 1, 0
714				
715			:ITEM 24	
716	001564	027337	EM24	:SUMMARY OF COUNT PATTERN FAILURES IN CACHE MAINTENANCE REGISTER
717	001566	035007	DH23	:EXPECTD EXPECTD RECEIVED RECEIVED
718				:OR AND OR AND #ERRORS TESTNO
719	001570	037212	DT23	:PATTOR,PATAND,DATAOR,DATAND,ERRCNT,TESTNO,0
720	001572	037671	DF23	: 0, 0, 0, 0, 1, 0
721				
722			:ITEM 25	
723	001574	027437	EM25	:REFERENCED MAP REGISTER 0 WITH ADDRESS ONE BIT
724				:DIFFERENT THAN 770200
725	001576	035126	DH25	:ADDRUSED BITDIFF TESTNO ERRORPC
726	001600	037230	DT25	:SREG4,SREG0,TESTNO,SERRPC,0
727	001602	037677	DF25	: 3, 4, 0, 0
728				
729			:ITEM 26	
730	001604	027544	EM26	:REFERENCED CACHE LOW ADDRESS REGISTER WITH
731				:ADDRESS ONE BIT DIFFERENT THAN 777740
732	001606	035126	DH25	:ADDRUSED BITDIFF TESTNO ERRORPC

733	001610	037230	DT25	: \$REG4, \$REG0, TESTNO, \$ERRPC, 0
734	001612	037677	DF25	: 3, 4, 0, 0
735				
736			: ITEM 27	
737	001614	027544	EM26	: REFERENCED CACHE LO ADDRESS REGISTER WITH
738				: ONE BIT DIFFERENT THAN 777740
739	001616	035170	DH27	: ADDRUSED TESTNO ERRORPC
740	001620	037242	DT27	: \$REG1, TESTNO, \$ERRPC
741	001622	037703	DF27	: 3, 0, 0
742				
743			: ITEM 30	
744	001624	027665	EM30	: CAN'T GET TO MAIN MEMORY FROM UNIBUS WITH THE MAP OFF
745				: SO I'LL JUMP TO THE SIZE JUMPER TEST FOR VERIFICATION
746	001626	035222	DH30	: TESTNO ERRORPC
747	001630	037252	DT30	: TESTNO, \$ERRPC, 0
748	001632	037706	DF30	: 0, 0
749				
750			: ITEM 31	
751	001634	030041	EM31	: SUMMARY OF COUNT PATTERN FAILURES ON THE UNIBUS DATA PATH
752	001636	035007	DH23	: EXPECTD EXPECTD RECEIVD RECEIVD
753				: 'OR' 'AND' 'OR' 'AND' #ERRORS TESTNO
754	001640	037212	DT23	: PATTOR, PATAND, DATAOR, DATAND, ERRCTN, TESTNO, 0
755	001642	037671	DF23	: 0, 0, 0, 0, 1, 0
756				
757			: ITEM 32	
758	001644	030133	EM32	: UNIBUS MAP IS RELOCATING WHEN NOT ENABLED
759	001646	035222	DH30	: TESTNO ERRORPC
760	001650	037252	DT30	: TESTNO, \$ERRPC, 0
761	001652	037706	DF30	: 0, 0
762				
763			: ITEM 33	
764	001654	030205	EM33	: CANNOT USE ANY OF THE MAP REGISTERS OR PHYSICAL
765				: ADDRESS BIT14 IS STUCK LOW, MUST RESTART PROGRAM
766				: IF YOU DON'T LOOP ON THIS PROBLEM.
767	001656	035222	DH30	: TESTNO ERRORPC
768	001660	037252	DT30	: TESTNO, \$ERRPC, 0
769	001662	037706	DF30	: 0, 0
770				
771			: ITEM 34	
772	001664	030362	EM34	: THE NUMBER OF MAP REGISTERS REMOVED BY JUMPER SETTING
773				: DOES NOT AGREE WITH THE NUMBER FOUND TO BE MISSING.
774	001666	035242	DH34	: REMOVED MISSING TESTNO ERRORPC
775	001670	037260	DT34	: ERRCNT, CNTR, TESTNO, \$ERRPC, 0
776	001672	037710	DF34	: 0, 0, 0, 0
777				
778			: ITEM 35	
779	001674	030533	EM35	: THE SIZE JUMPERS ON THE UNIBUS MAP ARE NOT SET
780				: IN THEIR DEFAULT POSITION. THIS WOULD ALLOW UNIBUS
781				: ADDRESSES 000000 TO 757776 TO REFERNECE MAIN MEMORY
782				: THEIR CURRENT SETTING ALLOWS ONLY:
783	001676	035302	DH35	: LOWEST HIGHEST TESTNO ERRORPC
784	001700	037272	DT35	: LOWEST, HIGEST, TESTNO, \$ERRPC, 0
785	001702	037714	DF35	: 4, 4, 0, 0
786				
787			: ITEM 36	
788	001704	031020	EM36	: MAP REGISTER UNDER TEST DID NOT RESPOND IN DUAL MAPPING TEST

789	001706	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
796				: 'OR' 'AND' 'OR' 'AND' #ERRORS TESTNO
797	001720	037314	DT37	:ADDROR,ADRAND,DATAOR,DATAAND,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
803				:EXPECTD RECEIVD TESTNO ERRORPC
804	001730	037332	DT40	:CPUEXP,PCPUER,TESTNO,\$ERRPC,0
805	001732	037731	DF40	: 0, 0, 0, 0
806				
807			:ITEM 41	
808	001734	031113	EM41	:RELOCATION THROUGH THE MAP WAS NOT CORRECT, FULL ADD.
809	001736	035514	DH41	:CORRECT ADDRESS
810				:ADDRESS FETCHED TESTNO ERRORPC
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
817				:ADDRESS DATA FROM UB TESTNO ERRORPC
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	:SIZJUMP 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
830				:EXPECTD RECEIVD REFERENC'D REGISTR REGISTR TESTNO ERRORPC
831	001770	037404	DT44	:STMP4,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
838				:EXPECTD RECEIVD TESTNO ERRORPC
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

845	002006	035514	DH41	:CORRECT ADDRESS
846				:ADDRESS FETCHED TESTNO ERRORPC
847	002010	037344	DT41	:\$REG2,\$REG1,TESTNO,\$ERRPC
848	002012	037735	DF41	: 0, 0, 0, 0
849				
850			:ITEM 47	
851	002014	032107	EM47	:RELOCATION THRU THE MAP WAS NOT CORRECT, CARRY PROPAGATION
852				:TEST BEING RUN OVER UNIBUS
853	002016	035574	DH42	:CORRECT EXPECTD RECEIVD
854				:ADDRESS DATA FROM UB TESTNO ERRORPC
855	002020	037356	DT42	:\$REG1,\$REG3,\$REG2,TESTNO,\$ERRPC
856	002022	037741	DF42	: 3, 0, 0, 0, 0
857				
858			:ITEM 50	
859	002024	032242	EM50	:THE TOP OF MEMORY IS DIFFERENT THAN THE SIZE JUMPERS
860				:TEST BEING RUN OVER UNIBUS
861	002026	035674	DH43	:SIZJUMP TOPFOUND TESTNO
862	002030	037372	DT43	:SIZELO,RSIZE,TESTNO,0
863	002032	037742	DF43	: 0, 0, 0
864				
865				
866			:ITEM 51	
867	002034	032367	EM51	:PARITY REPORTING THRU THE MAP IS NOT CORRECT
868				:TEST BEING RUN OVER UNIBUS
869	002036	035734	DH44	:CONDITN CONDITN ADDRESS MAINTEN CONTROL
870				:EXPECTD RECEIVD REFERENC'D REGISTR REGISTR TESTNO ERRORPC
871	002040	037404	DT44	:\$TMP4,\$PPARER,\$PLOADR,\$PMAINT,\$PCONTR,TESTNO,\$ERRPC,0
872	002042	037752	DF44	: 0, 0, 2, 0, 0, 0, 0
873				
874			:ITEM 52	
875	002044	032504	EM52	:SUMMARY OF DUAL MAPPING ERRORS
876	002046	035007	DH23	:EXPECTD EXPECTD RECEIVD RECEIVD
877				: 'OR' 'AND' 'OR' 'AND' #ERRORS TESTNO
878	002050	037314	DT37	:ADDROR,ADRAND,DATAOR,DATAND,\$ERRPC,TESTNO,0
879	002052	037723	DF37	: 0, 0, 0, 0, 1, 0
880				
881			:ITEM 53	
882				
883	002054	032543	EM53	:BYP BIT IN USMR COULD NOT BE CLEARED
884	002056	036100	DH53	
885	002060	037424	DT53	
886	002062	037706	DF30	
887				
888			:ITEM 54	
889				
890	002064	032610	EM54	:BYP BIT IN USMR COULD NOT BE SET
891	002066	036100	DH53	
892	002070	037424	DT53	
893	002072	037706	DF30	
894				
895			:ITEM 55	
896	002074	032651	EM55	:MK11 CSR COULD NOT BE ACCESSED
897	002076	036127	DH55	
898	002100	037434	DT55	
899	002102	037606	DF1	
900				

901			:ITEM 56	
902				:TEST DATA REF NOT A HIT
903	002104	032725	EM56	
904	002106	036172	DH56	
905	002110	037446	DT56	
906	002112	037644	DF12	
907				
908			:ITEM 57	
909				:TEST DATA REF NOT A MISS
910	002114	032763	EM57	:CACHE BYPASS ON UNIBUS MAP DID NOT INVALIDATE CACHED DATA
911	002116	036237	DH57	
912	002120	037460	DT57	
913	002122	037606	DF1	
914				
915	002124		ER200:	:THIS IS THE STARTING POINT FOR ERROR MESSAGES
916				:201 THRU 377. THEY ARE USED FOR MULTIPLE
917				:ERROR MESSAGES.
918				
919			:ITEM 201	
920	002124	033122	EM201	:THE FOLLOWING REGISTERS TIMED OUT WHEN READ
921	002126	036303	DH201	:REGADRS TESTNO ERRORPC
922	002130	037472	DT201	:SREG0,TESTNO,SERRPC,0
923	002132	037761	DF201	: 0, 0, 0
924				
925			:ITEM 202	
926	002134	033176	EM202	:THE FOLLOWING MAP REGISTERS WILL NOT CLEAR
927	002136	036333	DH202	:REGADRS DATAREC TESTNO ERRORPC.
928	002140	037502	DT202	:SREG0,SREG1,TESTNO,SERRPC,0
929	002142	037764	DF202	: 0, 0, 0, 0
930				
931			:ITEM 203	
932	002144	033251	EM203	:THE FOLLOWING ARE DUAL ADDRESSING ERRORS IN THE UNIBUS MAP
933	002146	036373	DH203	:MAPREG MAPREG
934				:TESTING DUALED TESTNO ERRORPC
935	002150	037514	DT203	:SREG0,SREG1,SREG2,TESTNO,SERRPC,0
936	002152	037770	DF203	: 0, 0, 0, 0
937				
938			:ITEM 204	
939	002154	033344	EM204	:THE COUNT PATTERN THRU THE MAP REGISTERS FAILED
940	002156	036452	DH204	:REGADRS PATTERN EXPECTD RECEIVD TESTNO ERRORPC
941	002160	037526	DT204	:SREG0,SREG2,SREG4,SREG3,TESTNO,SERRPC,0
942	002162	037774	DF204	: 0, 0, 0, 0, 0, 0
943				
944			:ITEM 205	
945	002164	033424	EM205	:UNIBUS DATA PATH COUNT PATTERN FAILURE
946	002166	036532	DH205	:EXPECTD RECEIVD ADDRLOAD TESTNO ERRORPC
947	002170	037544	DT205	:SREG1,SREG0,SREG2,TESTNO,SERRPC,0
948	002172	040002	DF205	: 0, 0, 3, 0, 0
949				
950			:ITEM 206	
951	002174	033473	EM206	:UNIBUS ADDRESSING ERRORS. MAP RELOCATION DISABLED
952	002176	036604	DH206	:ADDRESS ADDRESS
953				:EXPECTD RECEIVD TESTNO ERRORPC
954	002200	037560	DT206	:SREG0,SREG3,TESTNO,SERRPC,0
955	002202	040007	DF206	: 0, 0, 0, 0
956				

```
957 :ITEM 207
958 002204 033555 EM207 :COUNT PATTERN FAILURES IN CACHE REGISTERS
959 002206 036664 DM207 :REGISTR EXPECTD RECEIVD
960 :ADDRESS DATA DATA TESTNO ERRORPC
961 002210 037572 D1207 :$REG0,$REG2,$REG3,TESTNO,$ERRPC,0
962 002212 040013 DF207 : 0, 0, 0, 0, 0
963
964
965
966 ;:*****
967
968 .SBTTL SCOPE HANDLER ROUTINE
969
970 *THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
971 *AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
972 *AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
973 *THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
974 *SW14=1 LOOP ON TEST
975 *SW11=1 INHIBIT ITERATIONS
976 *SW09=1 LOOP ON ERROR
977 *SW08=1 LOOP ON TEST IN SWR<6:0>
978 *CALL
979 * SCOPE ;;SCOPE=10T
980
981 $SCOPE:
982 002214 005037 001322 CLR RETRY :CLEAR RETRY FLAG AN THE START OF
983 :EACH TEST
984 002220 005037 001254 CLR ERRCNT :CLEAR THE MULTIPLE ERROR COUNTER
985 002224 005037 001232 CLR DATAOR :LGCATION FOR LOGICAL OR OF BAD DATA
986 002230 005037 001226 CLR ADDROR :LOCATION FOR LOGICAL OR OF ADDRESS
987 002234 005037 001236 CLR PATTOR :LOCATION FOR LOGICAL OR OF PATTERN LOADED
988 002240 012700 177777 MOV #-1,R0 :LOAD -1 INTO R0 TO INITIALIZE LOGICAL AND LOCS
989 002244 010037 001230 MOV R0,DATAND :LOCATION FOR LOGICAL AND OF BAD DATA
990 002250 010037 001224 MOV R0,ADRAND :LOCATION FOR LOGICAL AND OF ADDRESS
991 002254 010037 001234 MOV R0,PATAND :LOCATION FOR LOGICAL AND OF PATTERN LOADED
992 002260 006137 177570 ROL @#SWR :LOOP ON PRESENT TEST?
993 002264 100514 BMI $OVER :YES IF SW14=1
994
995 002266 000416 :####START OF CODE FOR THE XOR TESTER####
996 $XTSTR: BR 6$ :IF RUNNING ON THE 'XOR' TESTER CHANGE
997 :THIS INSTRUCTION TO A 'NOP' (NOP=240)
998 002270 013746 000004 MOV @#ERPVEC,-(SP) :SAVE THE CONTENTS OF THE ERROR VECTOR
999 002274 012737 002314 000004 MOV #55,@#ERRVEC :SET FOR TIMEOUT
1000 002302 005737 177060 TST @#177060 :TIME OUT ON XOR?
1001 002306 012637 000004 MOV (SP)+,@#ERRVEC :RESTORE THE ERROR VECTOR
1002 002312 000466 BR $SYLAD :GO TO THE NEXT TEST
1003 002314 022626 5$: CMP (SP)+,(SP)+ :CLEAR THE STACK AFTER A TIME OUT
1004 002316 012637 000004 MOV (SP)+,@#ERRVEC :RESTORE THE ERROR VECTOR
1005 002322 000426 BR 7$ :LOOP ON THE PRESENT TEST
1006 002324 032737 000400 177570 6$:####END OF CODE FOR THE XOR TESTER####
1007 002332 001407 BIT #BIT08,@#SWR :LOOP ON SPEC. TEST?
1008 002334 013746 177570 BEQ 2$ :BR IF NO
1009 002340 042716 000200 MOV @#SWR,-(SP) :SET DESIRED TEST NUM. FROM SWR
1010 002344 122637 001102 BIC #5SWR&K,(SP) :STRIP AWAY UNDESIRED BITS
1011 002350 001462 CMPB (SP)+,$TSTNM :ON THE RIGHT TEST?
1012 002352 105737 001103 BEQ $OVER :BR IF YES
1013 2$: 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   $STNM     ;;COUNT TEST NUMBERS
1033 002474 011637 001106          MOV     (SP),$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  $JVER: MOV     $STNM,@#DISPLAY ;;DISPLAY TEST NUMBER
1038 002524 013716 001106          MOV     $LPADR,(SP) ;;FUDGE RETURN ADDRESS
1039 002530 000002          RTI
1040 002532 000144          SMXCNT: 100.        ;;FIXES PS
1041          ;;*****
1042          ;;*****
1043          ;;*****
1044          ;;*****
1045          ;;*****
1046          ;;*****
1047          ;;*****
1048          ;;*****
1049          ;;*****
1050          ;;*****
1051          ;;*****
1052          ;;*****
1053          ;;*****
1054          ;;*****
1055          ;;*****
1056          ;;*****
1057          ;;*****
1058          ;;*****
1059          ;;*****
1060          ;;*****
1061          ;;*****
1062          ;;*****
1063          ;;*****
1064          ;;*****
1065          ;;*****
1066          ;;*****
1067          ;;*****
1068          ;;*****

```

.SBTTL ERROR HANDLER ROUTINE

```

;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
;*AND GO TO ERTYPE ON ERROR
;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
;*SW15=1      HALT ON ERROR
;*           HALT CAN OCCUR BEFORE AND AFTER THE ERROR TYPEOUT
;*           *
;*SW13=1      INHIBIT ERROR TYPEOUTS
;*SW10=1      BELL ON ERROR
;*SW09=1      LOOP ON ERROR
;*CALL
;*           ERROR   N      ;;ERROR=EMT AND N=ERROR ITEM NUMBER

```

```

$ERROR:  MOVB   $STNM,$TESTNO ;;SAVE TEST NUMBER FOR ERROR TYPE OUT
          INC    ERRCNT     ;;COUNT ALL MULTIPLE ERRORS
          MOV    R0,$REG0   ;;SAVE R0 FOR POSSIBLE TYPE OUT
          MOV    R1,$REG1   ;;SAVE R1 FOR POSSIBLE TYPE OUT
          MCV   R2,$REG2   ;;SAVE R2 FOR POSSIBLE TYPE OUT
          MOV    R3,$REG3   ;;SAVE R3 FOR POSSIBLE TYPE OUT
          MOV    R4,$REG4   ;;SAVE R4 FOR POSSIBLE TYPE OUT
          MOV    R5,$REG5   ;;SAVE R5 FOR POSSIBLE TYPE OUT
7$:      INCB   $ERFLG     ;;SET THE ERROR FLAG
          BEQ    7$        ;;DON'T LET THE FLAG GO TO ZERO
          MOV    $STNM,@#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		TYPE	\$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	::STRIP AND SAVE THE ERROR ITEM CODE
1078	002654	117737	176236	001114	MOVB	@\$ERRPC,\$ITEMB	::SKIP TYPEOUT IF SET
1079	002662	032737	020000	177570	BIT	#BIT13,@#SWR	::SKIP TYPEOUTS
1080	002670	001004			BNE	2\$::GO TO USER ERROR ROUTINE
1081	002672	004737	003036		JSR	PC,ERTYPE	
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	#SENDAD,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\$: BIT	#SW9,SWR	::ARE YOU LOOPING ON THIS ERROR?
1096	002752	032737	001000	177570	BEQ	EREXIT	::BRANCH IF NOT LOOPING ON ERROR
1097	002760	001425			MOV	#-1,@#CPUERR	::CLEAR CPU ERROR REGISTER
1098	002762	012737	177777	177766	MOV	#-1,@#MEMERR	::CLEAR MEMORY ERROR REGISTER
1099	002770	012737	177777	177744	MOV	#-1,@#MEMERR	::CLEAR MEMORY ERROR REGISTER
1100	002776	042737	177776	177572	BIC	#177776,@#MPRO	::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,MPFLAG	::INITIALIZE MEMORY MANAGEMENT TRAP FLAG
1105	003034	000002			EREXIT: RTI		::RETURN TO TEST

.SBTTL ERROR MESSAGE TYPE OUT ROUTINE

1106							
1107							
1108							
1109							
1110							
1111	::*						
1112	::*						
1113	::*						
1114	::*						
1115	::*						
1116	::*						
1117	::*						
1118	::*						
1119	::*						
1120	::*						
1121	::*						
1122	::*						
1123	::*						
1124	::*						

THIS SUBROUTINE IS CALLED BY THE ERROR HANDLER TO TYPE THE ERROR MESSAGES. IT PICKS UP THE ITEM BYTE (\$ITEMB) NUMBER AND USES THAT TO INDEX THROUGH THE ERROR TABLE. THE ERROR TABLE STARTS AT '\$ERTTB' AND HAS FOUR (4) POINTERS FOR EACH ENTRY, 'EM', 'DH', 'DT', 'DF'. THE 'EM' POINTS TO THE ERROR MESSAGE WHICH IS AN ASCIZ STRING. THE 'DH' POINTS TO THE DATA HEADER WHICH IS ANOTHER ASCIZ STRING. THE 'DT' POINTS TO THE DATA TABLE WHICH IS A GROUP OF WORDS CONTAINING THE ADDRESSES OF THE DATA TO BE TYPED. THE FORMAT OF THIS DATA IS CONTROLLED BY THE 'DF' WHICH IS THE POINTER TO THE DATA FORMAT. THE DATA FORMAT IS A GROUP OF BYTES WHICH CONTAIN NUMBERS THAT CORRESPOND TO DIFFERENT TYPING FORMATS.

0	-16 BIT OCTAL FORMAT
1	-DECIMAL FORMAT

```

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      MOV      @#SITEMB,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 $CRLF' IF SW 7 IS UP
1149      : *      : *      ;INHIBIT MULTIPLE ERROR TYPEOUTS
1150      : *      : *      ;BRANCH TO EXIT
1151 003114 000507      BR      13$     ;BRANCH TO EXIT
1152 003116 042700 177000      20$: BIC    #177000,R0 ;CLEAR UPPER BYTE OF R0
1153 003122 062700 002130      ADD    #ER200+4,R0 ;POINT TO DATA TABLE ENTRY
1154 003126 000424      BR      5$      ;GO TYPE DATA TABLE
1155 003130 042700 177000      21$: BIC    #177000,R0 ;CLEAR UPPER BYTE OF R0
1156 003134 062700 000570      ADD    #<ER200-$ERRTB>,R0 ;ADD DIFFERENCE BETWEEN
1157      : *      : *      ;ITEM 1 AND ITEM 201
1158      : *      : *      GET POINTER TO ERROR MESSAGE AND TYPE IT
1159      : *      : *      IF THE POINTER IS NOT ZERO
1160 003140 062700 001334      22$: ADD    #ERRTB,R0 ;ADD BASE OF ERROR TABLE
1161 003144 012037 003154      MOV    (R0)+,2$ ;PUT MESSAGE POINTER IN TYPE STATEMENT
1162 003150 001404      BEQ    3$      ;BRANCH IF NO ERROR MESSAGE
1163 003152 104400      TYPE   ;TYPE ERROR MESSAGE
1164 003154 000000      2$: .WORD 0 ;POINTER TO ERROR MESSAGE
1165 003156 104400 001215      TYPE   .CRLF ;TYPE CRLF
1166      : *      : *      GET THE POINTER TO THE DATA HEADER AND
1167      : *      : *      TYPE IT IF THE POINTER IS NOT ZERO
1168 003162 012037 003172      3$: MOV    (R0)+,4$ ;PUT HEADER POINTER IN TYPE STATEMENT
1169 003166 001404      BEQ    5$      ;BRANCH IF NO DATA HEADER
1170 003170 104400      TYPE   ;TYPE THE DATA HEADER
1171 003172 000000      4$: .WORD 0 ;POINTER TO DATA HEADER
1172 003174 104400 001215      TYPE   .CRLF ;TYPE CRLF
1173      : *      : *      THIS IS THE START OF THE DATA OUTPUT IF THE
1174      : *      : *      DATA POINTER IS NOT ZERO. R0 POINTS TO THE
1175      : *      : *      DATA FORMAT, R1 POINTS TO THE ADDRESS OF
1176      : *      : *      THE DATA WORDS.
1177 003200 010146      5$: MOV    R1,-(KSP) ;SAVE R1 ON THE STACK
1178 003202 012001      MOV    (R0)+,R1 ;PUT DATA TABLE POINTER IN R1
1179 003204 001452      BEQ    12$     ;BRANCH IF NO DATA TABLE
1180 003206 012000      MOV    (R0)+,R0 ;PICK UP DATA FORMAT POINTER
1181 003210 105710      6$: TSTB (R0) ;IS THIS WORD OCTAL

```



```

1181 003212 001003      BNE      7$          ;BRANCH IF NOT 16-BIT OCTAL
1182      ::          ;THIS IS 16 BIT OCTAL FORMAT (DF = 0)
1183 003214 013146      MOV      @ (R1)+,-(KSP) ;PUT WORD ON STACK FOR TYPING
1184 003216 104402      TYPDC          ;TYPE THE WORD ON STACK AS 16 BIT OCTAL
1185 003220 000436      BR       11$        ;GET READY FOR NEXT WORD
1186 003222 122710 000001 7$:      CMPB    #1,(R0)    ;IS THE WORD DECIMAL
1187 003226 001003      BNE      8$          ;BRANCH IF NOT DECIMAL
1188      ::          ;THIS IS DECIMAL FORMAT (DF = 1)
1189 003230 013146      MOV      @ (R1)+,-(KSP) ;PUT WORD ON STACK FOR TYPING
1190 003232 104410      TYPDS          ;TYPE THE WORD ON STACK AS DECIMAL
1191 003234 000430      BR       11$        ;GET READY FOR NEXT WORD
1192 003236 122710 000002 8$:      CMPB    #2,(R0)    ;IS WORD 22-BIT PHYSICAL ADDRESS
1193 003242 001012      BNE      9$          ;BRANCH IF NOT 22-BIT PHYSICAL ADDR
1194      ::          ;THIS IS 22-BIT PHYSICAL FORMAT (DF = 2)
1195 003244 012146      MOV      (R1)+,-(KSP)  ;PUT ADDR OF LOW WORD ON STACK
1196 003246 004737 004750 JSR      PC,$DB20    ;CONVERT NUMBER TO OCTAL ASCIZ
1197 003252 062716 000003 ADD      #3,(KSP)    ;ONLY WANT 8 DIGITS
1198 003256 012637 003264 MOV      (KSP)+,30$   ;PUT POINTER AFTER 'TYPE' CALL
1199 003262 104400      TYPE          ;TYPE ASCIZ STRING
1200 003264 000000 30$:      .WORD    0          ;WORD HOLDS POINTER TO ASCIZ STRING
1201 003266 000413      BR       11$        ;GET READY FOR NEXT WORD
1202 003270 122710 000003 9$:      CMPB    #3,(R0)    ;IS THIS A 16-BIT VIRTUAL ADDRESS
1203 003274 001004      BNE      10$        ;BRANCH IF NOT 16-BIT VIRT. ADDR.
1204      ::          ;THIS IS 22-BIT VIRTUAL ADDRESS FORMAT
1205      ::          ;KERNEL 1-SPACE ASSUMED. (DF = 3)
1206 003276 013146      MOV      @ (R1)+,-(KSP) ;PUT 16-BIT VIRTUAL ADDR ON STACK
1207 003300 004737 003344 JSR      PC,TYPVAD   ;GO TYPE 22-BIT ADDRESS FROM 16-BIT V.A.
1208 003304 000404      BR       11$        ;GET READY FOR NEXT WORD
1209      ::          ;THIS IS FORMAT 4. DATA WORD IS A UNIBUS ADDRESS
1210      ::          ;OUTPUT WILL BE 18-BITS WORD LEFT SHIFTED 6.
1211 003306 013146 10$:      MOV      @ (R1)+,-(KSP) ;PUSH 16-BIT UNIBUS ADDRESS ON STACK
1212 003310 004737 003452 JSR      PC,UBADDR   ;CONVERT TO 18-BIT UNIBUS ADDR AND TYPE
1213 003314 000400      BR       11$        ;GET READY FOR NEXT WORD
1214 003316 005200 11$:      INC      R0          ;POINT TO NEXT FORMAT BYTE
1215 003320 104400 003340 TYPE      ,32$       ;TYPE TWO SPACES
1216 003324 005711      TST     (R1)        ;IS THERE ANOTHER WORD?
1217 003326 001401      BEQ     12$        ;BRANCH IF ALL DONE
1218 003330 000727      BR      6$          ;GO BACK FOR NEXT NUMBER
1219 003332 012601 12$:      MOV     (KSP)+,R1   ;RESTORE R1
1220 003334 012600 13$:      MOV     (KSP)+,R0   ;RESTORE R0
1221 003336 000207      RTS     PC          ;RETURN TO ERROR ROUTINE
1222 003340 020040 000 32$:      .ASCIZ  ? ?        ;TWO SPACES
1223      .EVEN
1224
1225
1226      .SBTTL  CONVERT 16-BIT VIRTUAL ADDRESS TO 22-BIT PHYSICAL ADDRESS
1227      *
1228      *      THIS ROUTINE IS CALLED BY A 'JSR PC' AFTER THE VIRTUAL ADDRESS
1229      *      IS PUSHED ON THE KERNEL STACK. THE V.A. IS THEN LOADED INTO
1230      *      R1 AND THE UPPER 3 BITS ARE SHIFTED INTO R0 TO SELECT THE
1231      *      CORRECT KERNEL 1-SPACE PAR. THE LOWER 12 BITS OF THE VIRTUAL
1232      *      ADDRESS ARE ADDED TO THE PAR AS THEY ARE BY MEMORY MANAGEMENT
1233      *      AND THE PHYSICAL ADDRESS IS SAVED IN MEMORY TO BE CONVERTED
1234      *      TO ASCIZ AND TYPED.
1235      *
1236 003344 104412      TYPVAD: SAVREG      ;SAVE ALL REGISTERS

```

1237 003346 016601 000002
 1238 003352 005000
 1239 003354 073027 000003
 1240 003360 006300
 1241 003362 006001
 1242 003364 006001
 1243 003366 006001
 1244 003370 062700 172340
 1245 003374 011003
 1246 003376 005002
 1247 003400 073227 000006
 1248 003404 060103
 1249 003406 005502
 1250 003410 010237 001222
 1251 003414 010337 001220
 1252 003420 012746 001220
 1253 003424 004737 004750
 1254 003430 062716 000003
 1255 003434 012637 003442
 1256 003440 104400
 1257 003442 000000
 1258
 1259 003444 104414
 1260 003446 012616
 1261 003450 000207
 1262
 1263
 1264
 1265
 1266
 1267
 1268 003452 016601 000002
 1269 003456 005000
 1270 003460 073027 000006
 1271 003464 010137 001220
 1272 003470 010037 001222
 1273 003474 012746 001220
 1274 003500 004737 004750
 1275 003504 062716 000005
 1276 003510 012637 003516
 1277 003514 104400
 1278 003516 000000
 1279 003520 012616
 1280 003522 000207
 1281
 1282
 1283
 1284
 1285
 1286
 1287
 1288
 1289
 1290
 1291
 1292

```

MOV 2(KSP),R1      ;PUT VIRTUAL ADDR IN R1
CLR R0             ;CLEAR R0 FOR CALCULATIONS
ASHC #3,R0        ;LEFT SHIFT R0,R1 3 PLACES
ASL R0            ;LEFT SHIFT R0 ONE MORE PLACE
ROR R1            ;RIGHT SHIFT R1 SO OFFSET IS CORRECT
ROR R1            ;RIGHT SHIF R1
ROR R1            ;RIGHT SHIFT R1
ADD #KIPAR0,R0    ;FORM DESIRED PAR ADDR IN R0
MOV (R0),R3       ;PUT CONTENTS OF PAR IN R3
CLR R2            ;CLEAR R2 FOR PHYSICAL ADDR CALCULATIONS
ASHC #6,R2        ;LEFT SHIFT <R2,R3> 6 PLACES
ADD R1,R3         ;ADD OFFSET IN R1 TO BASE IN R3
ADC R2            ;ADD ANY POSSIBLE CARRY TO UPPER 6 BITS
MOV R2,PADRSH    ;PUT UPPER 6 BITS OF ADDR IN CORE
MOV R3,PADRSL    ;PUT LOWER 16 BITS OF ADDR IN CORE
MOV #PADRSL,-(KSP) ;PUT POINTER TO LOWER 16 BITS ON STACK
JSR PC,$DB20     ;CONVERT NUMBER TO OCTAL ASCII
ADD #3,(KSP)      ;ONLY TYPE 8 DIGITS
MOV (KSP)+,3$    ;PUT POINTER AFTER TYPE INST
TYPE             ;TYPE THE 22-BIT VIRTUAL ADDRESS
3$: .WORD 0       ;THIS WORD HOLDS THE POINTER TO
                    ;THE ASCII STRING
RESREG           ;RESTORE ALL THE REGISTERS
MOV (KSP)+,(KSP) ;LEAVE ONLY RETURN ADDR ON STACK
RTS PC           ;RETURN TO ERROR HANDLER
    
```

;*THIS SUBROUTINE IS USED TO CONVERT THE A WORD PUSHED
 ;*ON THE STACK INTO A UNIBUS ADDRESS AND TYPE IT AS A
 ;*6 DIGIT NUMBER. IT USES R1 & R0 AND LEAVES
 ;*ALL OTHER REGISTERS UNCHANGED.

```

UBADDR: MOV 2(KSP),R1      ;LOAD 16 BIT ADDRESS INTO R1
CLR R0             ;CLEAR R0 FOR CALCULATIONS
ASHC #6,R0        ;LEFT SHIFT <R0,R1> 6 PLACES
MOV R1,PADRSL    ;PUT LOWER 16 BITS IN PADRSL
MOV R0,PADRSH    ;PUT UPPER 6 BITS IN PADRSH
MOV #PADRSL,-(KSP) ;PUSH POINTER TO WORDS ON STACK
JSR PC,$DB20     ;JUMP TO CONVERT ROUTINE
ADD #5,(KSP)      ;ONLY USE LOWER 6 CHARS.
MOV (KSP)+,3$    ;PUT POINTER AFTER TYPE CALL.
TYPE             ;
3$: .WORD 0       ;HOLDS POINTER TO FIRST CHAR.
MOV (KSP)+,(KSP) ;LEAVE ONLY RETURN ADDRESS ON STACK
RTS PC           ;RETURN TO ERROR TYPE ROUTINE.
    
```

;*****

.SBTTL SAVE AND RESTORE R0-R5 ROUTINES

```

;*SAVE R0-R5
;*CALL:
;* SAVREG
;*UPON RETURN FROM $SAVREG THE STACK WILL LOOK LIKE:
;*
;*TOP---(+16)
    
```

1293
 1294
 1295
 1296
 1297
 1298
 1299
 1300
 1301 003524
 1302 003524 010046
 1303 003526 010146
 1304 003530 010246
 1305 003532 010346
 1306 003534 010446
 1307 003536 010546
 1308 003540 016646 000022
 1309 003544 016646 000022
 1310 003550 016646 000022
 1311 003554 016646 000022
 1312 003560 000002
 1313
 1314
 1315
 1316
 1317 003562
 1318 003562 012666 000022
 1319 003566 012666 000022
 1320 003572 012666 000022
 1321 003576 012666 000022
 1322 003602 012605
 1323 003604 012604
 1324 003606 012603
 1325 003610 012602
 1326 003612 012601
 1327 003614 012600
 1328 003616 000002
 1329
 1330
 1331
 1332
 1333
 1334
 1335
 1336
 1337
 1338
 1339
 1340
 1341
 1342
 1343
 1344
 1345
 1346
 1347
 1348

;*+2---(+18)
 :*+4---R5
 :*+6---R4
 :*+8---R3
 :*+10---R2
 :*+12---R1
 :*+14---R0

SSAVREG:

```

MOV R0,-(SP)      ;; PUSH R0 ON STACK
MOV R1,-(SP)      ;; PUSH R1 ON STACK
MOV R2,-(SP)      ;; PUSH R2 ON STACK
MOV R3,-(SP)      ;; PUSH R3 ON STACK
MOV R4,-(SP)      ;; PUSH R4 ON STACK
MOV R5,-(SP)      ;; PUSH R5 ON STACK
MOV 22(SP),-(SP)  ;; SAVE PS OF MAIN FLOW
MOV 22(SP),-(SP)  ;; SAVE PC OF MAIN FLOW
MOV 22(SP),-(SP)  ;; SAVE PS OF CALL
MOV 22(SP),-(SP)  ;; SAVE PC OF CALL
RTI
  
```

;*RESTORE R0-R5

;*CALL:

RESREG

\$RESREG:

```

MOV (SP)+,22(SP)  ;; RESTORE PC OF CALL
MOV (SP)+,22(SP)  ;; RESTORE PS OF CALL
MOV (SP)+,22(SP)  ;; RESTORE PC OF MAIN FLOW
MOV (SP)+,22(SP)  ;; RESTORE PS OF MAIN FLOW
MOV (SP)+,R5      ;; POP STACK INTO R5
MOV (SP)+,R4      ;; POP STACK INTO R4
MOV (SP)+,R3      ;; POP STACK INTO R3
MOV (SP)+,R2      ;; POP STACK INTO R2
MOV (SP)+,R1      ;; POP STACK INTO R1
MOV (SP)+,R0      ;; POP STACK INTO R0
RTI
  
```

.SBTTL TYPE ROUTINE

```

;*ROUTINE TO TYPE ASCII MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
;*NOTE1: $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
;*NOTE2: $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
;*NOTE3: $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
;*
;*CALL:
;*1) USING A TRAP INSTRUCTION
;*   TYPE ,MESADR      ;;MESADR IS FIRST ADDRESS OF AN ASCII STRING
;*OR
;*   TYPE
;*   MESADR
;*2) USING A JSR INSTRUCTION
;*   MOV PS,-(SP)     ;;PUSH PROCESSOR STATUS WORD ON THE STACK
;*   JSR PC,$TYPE     ;;CALL TYPE ROUTINE
  
```

```

1349      ;*      MESADDR      ;;FIRST ADRESS OF MESSAGE
1350
1351 003620 105737 001151 $TYPE: TSTB $TFPLG      ;;IS THERE A TERMINAL?
1352 003624 100002      BPL      1$      ;;BR IF YES
1353 003626 000000      HALT      ;;HALT HERE IF NO TERMINAL
1354 003630 000407      BR      3$      ;;LEAVE
1355 003632 010046      1$: MOV      RO,-(SP)      ;;SAVE RO
1356 003634 017600 000002      MOV      @2(SP),RO      ;;GET ADDRESS OF ASCIZ STRING
1357 003640 112046      2$: MOV      (RO)+,-(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)+,RO      ;;RESTORE RO
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 <H?>
1364 003662 001426      BEQ      8$      ;;BRANCH IF NOT
1365 003664 122716 000200      CMFB      #CRLF,(SP)
1366 003670 001004      BNE      5$      ;;POP <CR><LF> EQUIV
1367 003672 005726      TST      (SP)+
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      $CHARCNT      ;;DON'T COUNT THE NULL AS A CHARACTER
1379 C13736 000770      BR      7$      ;;LOOP
1380
1381      ;;HORIZONTAL TAB PROCESSOR
1382
1383 003740 112716 000040      8$: MOV      #' ,(SP)      ;;REPLACE TAB WITH SPACE
1384 003744 004737 003764      9$: 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      @STPS      ;;WAIT UNTIL PRINTER IS READY
1390 003770 100375      BPL      $TYPEC
1391 003772 116677 000002 175144      MOV      2(SP),@STPB      ;;LOAD CHAR TO BE TYPED INTO DATA REG.
1392 004000 122766 000015 000002      CMFB      #CR,2(SP)      ;;BRANCH IF
1393 004006 001003      BNE      1$      ;;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

```

1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428 004034 017646 000000
1429 004040 116637 000001 004257
1430 004046 112637 004261
1431 004052 062716 000002
1432 004056 000406
1433 004060 112737 000001 004257
1434 004066 112737 000006 004261
1435 004074 112737 000005 004256
1436 004102 010346
1437 004104 010446
1438 004106 010546
1439 004110 113704 004261
1440 004114 005404
1441 004116 062704 000006
1442 004122 110437 004260
1443 004126 113704 004257
1444 004132 016605 000012
1445 004136 005003
1446 004140 006105 1\$:
1447 004142 000404 BR 3\$:
1448 004144 006105 2\$:
1449 004146 006105 ROL R5
1450 004150 006105 ROL R5
1451 004152 010503 MOV R5,R3
1452 004154 006103 3\$:
1453 004156 105337 004260 DECB R3
1454 004162 100016 BPL 7\$:
1455 004164 042703 177770 BIC #177770,R3
1456 004170 001002 BNE 4\$:
1457 004172 005704 TST R4
1458 004174 001403 BEQ 5\$:
1459 004176 005204 4\$:
1460 004200 052703 000060 BIS #'0,R3

```
;;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    N              ;;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    N              ;;CALL FOR TYPEOUT
*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
*$CALL:
*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
*      TYPOC    N              ;;CALL FOR TYPEOUT
*$TYPOS: MOV      @ (SP),-(SP)  ;;PICKUP THE MODE
        MOV      1(SP),SOFILL  ;;LOAD ZERO FILL SWITCH
        MOV      (SP)+,SOMODE+1 ;;NUMBER OF DIGITS TO TYPE
        ADD     #2,(SP)        ;;ADJUST RETURN ADDRESS
        BR      $TYPON
*$TYPOC: MOV      #1,SOFILL    ;;SET THE ZERO FILL SWITCH
        MOV      #6,SOMODE+1   ;;SET FOR SIX(6) DIGITS
*$TYPON: MOV      #5,SOCNT     ;;SET THE ITERATION COUNT
        MOV      R3,-(SP)      ;;SAVE R3
        MOV      R4,-(SP)      ;;SAVE R4
        MOV      R5,-(SP)      ;;SAVE R5
        MOV      SOMODE+1,R4    ;;GET THE NUMBER OF DIGITS TO TYPE
        NEG     R4              ;;SUBTRACT IT FOR MAX. ALLOWED
        ADD     #6,R4          ;;SAVE IT FOR USE
        MOV      R4,SOMODE     ;;GET THE ZERO FILL SWITCH
        MOV      SOFILL,R4     ;;PICKUP THE INPUT NUMBER
        MOV      12(SP),R5     ;;CLEAR THE OUTPUT WORD
        CLR     R3            ;;ROTATE MSB INTO 'C'
1$:     ROL     R5              ;;GO DO MSB
        BR      3$            ;;FORM THIS DIGIT
2$:     ROL     R5
        ROL     R5
        MOV     R5,R3
3$:     ROL     R3            ;;GET LSB OF THIS DIGIT
        DECB   SOMODE         ;;TYPE THIS DIGIT?
        BPL    7$            ;;BR IF NO
        BIC   #177770,R3     ;;GET RID OF JUNK
        BNE   4$            ;;TEST FOR 0
        TST   R4             ;;SUPPRESS THIS 0?
        BEQ   5$            ;;BR IF YES
4$:     INC    R4             ;;DON'T SUPPRESS ANYMORE 0'S
        BIS   #'0,R3        ;;MAKE THIS DIGIT ASCII
```

```

1461 004204 052703 000040 5$: BIS #',R3 ::MAKE ASCII IF NOT ALREADY
1462 004210 110337 004254 MOV B R3,R5 ::SAVE FOR TYPING
1463 004214 104400 004254 TYPE 8$ ::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 $TYPDS:
1494 004262 010046 MOV R0,-(SP) ::PUSH R0 ON STACK
1495 004264 010146 MOV R1,-(SP) ::PUSH R1 ON STACK
1496 004266 010246 MOV R2,-(SP) ::PUSH R2 ON STACK
1497 004270 010346 MOV R3,-(SP) ::PUSH R3 ON STACK
1498 004272 010546 MOV R5,-(SP) ::PUSH R5 ON STACK
1499 004274 012746 020200 MOV #20200,-(SP) ::SET BLANK SWITCH AND SIGN
1500 004300 016605 000020 MOV 20(SP),R5 ::GET THE INPUT NUMBER
1501 004304 100004 BPL 1$ ::BR IF INPUT IS POS.
1502 004306 005405 NEG R5 ::MAKE THE BINARY NUMBER POS.
1503 004310 112766 000055 000001 MOV B #'-,1(SP) ::MAKE THE ASCII NUMBER NEG.
1504 004316 005000 1$: CLR R0 ::ZERO THE CONSTANTS INDEX
1505 004320 012703 004476 MOV #SDBLK,R3 ::SETUP THE OUTPUT POINTER
1506 004324 112723 000040 MOV B #',(R3)+ ::SET THE FIRST CHARACTER TO A BLANK
1507 004330 005002 2$: 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
1518 004360 106316
1519 004362 103003
1520 004364 116663 000001 177777
1521 004372 052702 000060
1522 004376 052702 000040
1523 004402 110223
1524 004404 005720
1525 004406 020027 000010
1526 004412 002746
1527 004414 003002
1528 004416 010502
1529 004420 000764
1530 004422 105726
1531 004424 100003
1532 004426 116663 177777 177776
1533 004434 105013
1534 004436 012605
1535 004440 012603
1536 004442 012602
1537 004444 012601
1538 004446 012600
1539 004450 104400 004476
1540 004454 016666 000002 000004
1541 004462 012616
1542 004464 000002
1543 004466 023420
1544 004470 001750
1545 004472 000144
1546 004474 000012
1547 004476 000004
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557 004506 010046
1558 004510 016600 000002
1559 004514 005740
1560 004516 111000
1561 004520 016000 004526
1562 004524 000200
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572 004526

    BMI 7$          ;;BR IF YES
    ASLB (SP)       ;;MSD?
    BCC 6$          ;;BR IF NO
    MOVB 1(SP),-1(R3) ;;YES--SET THE SIGN
    BIS #'0,R2     ;;MAKE THE BCD DIGIT ASCII
    BIS #' ,R2     ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
    MOVB R2,(R3)+  ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
    TST (R0)+      ;;JUST INCREMENTING
    CMP R0,#10     ;;CHECK THE TABLE INDEX
    BLT 2$         ;;GO DO THE NEXT DIGIT
    BGT 8$         ;;GO TO EXIT
    MOV R5,R2      ;;GET THE LSD
    BR 6$          ;;GO CHANGE TO ASCII
    TSTB (SP)+     ;;WAS THE LSD THE FIRST NON-ZERO?
    BPL 9$         ;;BR IF NO
    MOVB -1(SP),-2(R3) ;;YES--SET THE SIGN FOR TYPING
    CLRB (R3)      ;;SET THE TERMINATOR
    MOV (SP)+,R5   ;;POP STACK INTO R5
    MOV (SP)+,R3   ;;POP STACK INTO R3
    MOV (SP)+,R2   ;;POP STACK INTO R2
    MOV (SP)+,R1   ;;POP STACK INTO R1
    MOV (SP)+,R0   ;;POP STACK INTO R0
    TYPE $DBLK     ;;NOW TYPE THE NUMBER
    MOV 2(SP),4(SP) ;;ADJUST THE STACK
    MOV (SP)+,(SP)
    RTI           ;;RETURN TO USER

SDTBL: 10000.
       1000.
       100.
       10.
$DBLK: .BLKW 4
;:*****

.SBTTL TRAP DECODER
;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE 'TRAP' INSTRUCTION
;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
;*GO TO THAT ROUTINE.
$TRAP: MOV R0,-(SP) ;;SAVE R0
       MOV 2(SP),R0 ;;GET TRAP ADDRESS
       TST -(R0)    ;;BACKUP BY 2
       MOVB (R0),R0 ;;GET RIGHT BYTE OF TRAP
       MOV $TRPAD(R0),R0 ;;INDEX TO TABLE
       RTS R0       ;;GO TO ROUTINE

.SBTTL TRAP TABLE
;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
;*BY THE 'TRAP' INSTRUCTION.
: ROUTINE
: -----
$TRPAD:

```

1573	004526	003620			\$TYPE	::CALL=TYPE	TRAP+0(104400)	TTY TYPEOUT ROUTINE
1574	004530	004060			\$TYPOC	::CALL=TYPOC	TRAP+2(104402)	TYPE OCTAL NUMBER (WITH LEADING ZEROS)
1575	004532	004034			\$TYPOS	::CALL=TYPOS	TRAP+4(104404)	TYPE OCTAL NUMBER (NO LEADING ZEROS)
1576	004534	004074			\$TYPON	::CALL=TYPON	TRAP+6(104406)	TYPE OCTAL NUMBER (AS PER LAST CALL)
1577	004536	004262			\$TYPDS	::CALL=TYPDS	TRAP+10(104410)	TYPE DECIMAL NUMBER (WITH SIGN)
1578	004540	003524			\$SAVREG	::CALL=SAVREG	TRAP+12(104412)	SAVE R0-R5 ROUTINE
1579	004542	003562			\$RESREG	::CALL=RESREG	TRAP+14(104414)	RESTORE R0-R5 ROUTINE
1580	004544	005070			TBITOFF	::CALL=TBITO	TRAP+16(104416)	THIS WILL TURN OFF T BIT TRAPPING
1581	004546	005116			TBITRESTORE	::CALL=TBITR	TRAP+20(104420)	THIS WILL RETURN THE T BIT TO PR
1582								
1583								
1584								
1585								
1586								
1587								
1588	004550	012737	004676	000024	\$PWRDN:	MOV	#\$ILLUP,@PWRVEC	::SET FOR FAST UP
1589	004556	012737	000340	000026		MOV	#340,@PWRVEC+2	::PRIO:7
1590	004564	010046				MOV	R0,-(SP)	::PUSH R0 ON STACK
1591	004566	010146				MOV	R1,-(SP)	::PUSH R1 ON STACK
1592	004570	010246				MOV	R2,-(SP)	::PUSH R2 ON STACK
1593	004572	010346				MOV	R3,-(SP)	::PUSH R3 ON STACK
1594	004574	010446				MOV	R4,-(SP)	::PUSH R4 ON STACK
1595	004576	010546				MOV	R5,-(SP)	::PUSH R5 ON STACK
1596	004600	010637	004702			MOV	SP,\$SAVR6	::SAVE SP
1597	004604	012737	004616	000024		MOV	#\$PWRUP,@PWRVEC	::SET UP VECTOR
1598	004612	000000				HALT		
1599	004614	000776				BR	.-2	::HANG UP
1600								
1601								
1602	004616	013706	004702					
1603	004622	005037	004702					
1604	004626	005237	004702					
1605	004632	001375						
1606	004634	012605						
1607	004636	012604						
1608	004640	012603						
1609	004642	012602						
1610	004644	012601						
1611	004646	012600						
1612	004650	012737	004550	000024				
1613	004656	012737	000340	000026				
1614	004664	104400						
1615	004666	004704						
1616	004670	012716						
1617	004672	010000						
1618	004674	000002						
1619	004676	000000						
1620	004700	000776						
1621	004702	000000						
1622	004704	006412	047520	042527				
1623	004712	020122	040506	046111				
1624	004720	051125	026105	051040				
1625	004726	051505	040524	052122				
1626	004734	047111	020107	051120				
1627	004742	043517	040522	000115				
1628								

::*****

.SBTTL POWER DOWN AND UP ROUTINES

:POWER DOWN ROUTINE

```

$PWRDN: MOV    #$ILLUP,@PWRVEC  ::SET FOR FAST UP
        MOV    #340,@PWRVEC+2  ::PRIO:7
        MOV    R0,-(SP)        ::PUSH R0 ON STACK
        MOV    R1,-(SP)        ::PUSH R1 ON STACK
        MOV    R2,-(SP)        ::PUSH R2 ON STACK
        MOV    R3,-(SP)        ::PUSH R3 ON STACK
        MOV    R4,-(SP)        ::PUSH R4 ON STACK
        MOV    R5,-(SP)        ::PUSH R5 ON STACK
        MOV    SP,$SAVR6       ::SAVE SP
        MOV    #$PWRUP,@PWRVEC  ::SET UP VECTOR
        HALT
        BR     .-2             ::HANG UP
  
```

:POWER UP ROUTINE

```

$PWRUP: MOV    $SAVR6,SP       ::GET SP
        CLR    $SAVR6         ::WAIT LOOP FOR THE TTY
1$:     INC    $SAVR6         ::WAIT FOR THE INC
        BNE   1$              ::OF WORD
        MOV   (SP)+,R5        ::POP STACK INTO R5
        MOV   (SP)+,R4        ::POP STACK INTO R4
        MOV   (SP)+,R3        ::POP STACK INTO R3
        MOV   (SP)+,R2        ::POP STACK INTO R2
        MOV   (SP)+,R1        ::POP STACK INTO R1
        MOV   (SP)+,R0        ::POP STACK INTO R0
        MOV   #$PWRDN,@PWRVEC  ::SET UP THE POWER DOWN VECTOR
        MOV   #340,@PWRVEC+2  ::PRIO:7
        TYPE   PWRMSG         ::REPORT THE POWER FAILURE
        SPWRMG: .WORD PWRMSG  ::POWER FAIL MESSAGE POINTER
        MOV   (PC)+,(SP)     ::RESTART AT START
        SPWRAD: .WORD START   ::RESTART ADDRESS
        RTI
$ILLUP: HALT
        BR     .-2
$SAVR6: 0
PWRMSG: .ASCIZ <12><15>?POWER FAILURE, RESTARTING PROGRAM?
  
```

.EVEN

1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642 004750 104412
1643 004752 016601 000002
1644 004756 012705 005067
1645 004762 012704 000014
1646 004766 012703 177770
1647 004772 012100
1648 004774 012101
1649 004776 005002
1650 005000 110245
1651 005002 010002
1652 005004 005304
1653 005006 003007
1654 005010 001405
1655 005012 005205
1656 005014 010566 000002
1657 005020 104414
1658 005022 000207
1659 005024 006203
1660 005026 006001
1661 005030 006000
1662 005032 006001
1663 005034 006000
1664 005036 006001
1665 005040 006000
1666 005042 040302
1667 005044 062702 000060
1668 005050 000753
1669 005052 000016
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684

```
;;*****  
.SBTTL DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE  
;*THIS ROUTINE WILL CONVERT A 32-BIT UNSIGNED BINARY NUMBER TO AN  
;*UNSIGNED OCTAL ASCII NUMBER.  
;*CALL  
;*      MOV      #PNTR, -(SP)      ;; POINTER TO LOW WORD OF BINARY NUMBER  
;*      JSR      PC, @#SDB20      ;; CALL THE ROUTINE  
;*      RETURN                      ;; THE ADDRESS OF THE FIRST ASCII CHAR. IS ON THE STACK  
  
SDB20: SAVREG                      ;; SAVE ALL REGISTERS  
      MOV      2(SP), R1          ;; PICKUP THE POINTER TO LOW WORD  
      MOV      #SOCTVL+13., R5    ;; POINTER TO DATA TABLE  
      MOV      #12., R4          ;; DO ELEVEN CHARACTERS  
      MOV      #C7, R3           ;; MASK  
      MOV      (R1)+, R0         ;; LOWER WORD  
      MOV      (R1)+, R1         ;; HIGH WORD  
      CLR      R2                ;; TERMINATOR  
1$:   MOV      R2, -(R5)         ;; PUT CHARACTER IN DATA TABLE  
      MOV      R0, R2           ;; GET THIS DIGIT  
      DEC      R4               ;; COUNT THIS CHARACTER  
      BGT     3$, R4           ;; BR IF NOT THE LAST DIGIT  
      BEQ     2$, R4           ;; BR IF IT IS THE LAST DIGIT  
      INC      R5               ;; ALL DIGITS DONE-ADJUST POINTER FOR FIRST  
      MOV      R5, 2(SP)        ;; ASCII CHAR. & PUT IT ON THE STACK  
      RESREG                      ;; RESTORE ALL REGISTERS  
      RTS      PC               ;; RETURN TO USER  
2$:   ASR      R3                ;; POSITION THE MASK FOR THE LAST DIGIT  
3$:   ROR      R1                ;; POSITION THE BINARY NUMBER FOR  
      ROR      R0                ;; THE NEXT OCTAL DIGIT  
      ROR      R1  
      ROR      R0  
      ROR      R1  
      ROR      R0  
      BIC      R3, R2           ;; MASK OUT ALL JUNK  
      ADD      #0, R2           ;; MAKE THIS CHAR. ASCII  
      BR      1$, R2           ;; GO PUT IT IN THE DATA TABLE  
SOCTVL: .BLKB 14.             ;; RESERVE DATA TABLE
```

.SBTTL ***** SUBROUTINES UNIQUE TO THIS PROGRAM *****

```
.SBTTL TURN OFF AND SAVE T-BIT  
;*****  
;* THIS SUBROUTINE IS REACHED BY THE TRAP CALL 'TBITO'. IT IS  
;* USED TO TURN OFF THE T-BIT IF IT IS ON. THE PROCESSOR STATUS  
;* IS SAVED IN 'OLDPSW' SO THAT THE T-BIT CAN BE RESTORED TO ITS  
;* PREVIOUS STATUS WHEN CONDITIONS WARRANT.  
;*****
```

1685
1686 005070
1687 005070 032766 000020 000002
1688 005076 001406
1689 005100 016637 000002 001310
1690 005106 042766 000020 000002
1691 005114 000006
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705 005116
1706 005116 013766 001310 000002
1707 005124 042737 000020 001310
1708
1709 005132 000006
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722 005134 012703 170200
1723 005140 005023
1724 005142 022703 170376
1725 005146 103374
1726 005150 000207
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737 005152
1738 005152 005227
1739 005154 177777
1740 005156 001401

.EQUIV BIT4,TBIT ;T-BIT IS BIT04 IN PROC. STATUS
TBITOFF: BIT #TBIT,2(KSP) ;IS THE T-BIT ON?
BEQ 1\$;BRANCH TO EXIT IF IT IS NOT ON
MOV 2(KSP),OLDPSW ;SAVE OLD PSW FOR RESTORING T BIT
BIC #TBIT,2(KSP) ;CLEAR T BIT IF IT IS ON
1\$: RTT ;RETURN TO PROGRAM

.SBTTL RESTORE T-BIT TO ITS PREVIOUS CONDITION

* THIS SUBROUTINE CAN BE REACHED BY THE TRAP CALL 'TBITR', IT IS
* USED TO RESTORE THE T-BIT AFTER A PARTICULAR TEST THAT CANNOT
* BE RUN WITH THE T-BIT ON. IT USES THE PROCESSOR STATUS STORED
* IN 'OLDPSW' BY 'TBITO', REPLACES THE PS ON THE STACK WITH IT
* AND DOES AN 'RTT'.

TBITRESTORE:
MOV OLDPSW,2(KSP) ;PUT OLD PSW ON STACK
BIC #TBIT,OLDPSW ;CLEAR T-BIT IN 'OLDPSW' SO THAT
;IT WON'T BE TURNED ON BY ACCIDENT
RTT ;RETURN TO PROGRAM AND INHIBIT
;T BIT TRAP AFTER THIS INSTRUCTION.

.SBTTL SUBROUTINE TO CLEAR ALL OF THE MAP REGISTERS

* THIS SUBROUTINE CLEARS ALL OF THE MAP REGISTERS BY LOADING
* THE ADDRESS OF MAPLOO INTO R3 AND THEN CLEARING THE
* REGISTER POINTED TO BY R3 UNTIL R3 POINTS ABOVE MAPH37.

CLRMAP: MOV #MAPLO,R3 ;PUT FIRST MAP ADDR IN R3
1\$: CLR (R3)+ ;CLEAR MAP REGS
CMP #MAPH37,R3 ;SEE IF LAST ADDR IS IN R3
BHS 1\$;BRANCH IF R3 LE THAN LAST ADDR
RTS PC ;RETURN TO MAIN PROGRAM

.SBTTL SUBROUTINE TO LOG AND REPORT TIMEOUTS OF MAP REGISTERS

* THIS SUBROUTINE IS USED TO LOG AND REPORT THE FACT THAT A
* REFERENCE TO A MAPPING REGISTER TIMED OUT ON THE UNIBUS. IT
* KEEPS A 'LOGICAL AND' AND A 'LOGICAL OR' OF EACH ADDRESS THAT
* TIMES OUT.

TIMEOUT: ;STARTING ADDRESS OF SUBROUTINE
INC (PC)+ ;INCREMENT ONE TIME GATE
TOFLAG: .WORD -1 ;ONE TIME ENTRANCE FLAG
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$:  MOV      (KSP)+,OLDPC  ;SAVE RETURN ADDRESS
1747 005166 012637 001306    MOV      (KSP)+,OLDPS  ;SAVE OLD PSW
1748 005172 013737 177766    001266  MOV      (CPUERR,PCPUER ;SAVE CPU ERROR REGISTER
1749 005200 013737 001266    177766  MOV      PCPUER,CPUERR ;CLEAR CPU ERROR REGISTER
1750 005206 023737 001266    001264  CMP      PCPUER,CPUEXP ;SEE IF EXPECTED CONDITION CAME UP.
1751 005214 001402          BEQ      1$           ;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$:  BIS      RO,ADDROR   ;PERFORM LOGICAL OR OF FAILING ADDRESS
1755 005226 005100          COM      RO           ;GET RO READY FOR AND
1756 005230 040037 001224          BIC      RO,ADRAND   ;PERFORM LOGICAL AND
1757 005234 005100          COM      RO           ;PUT RO BACK AS IT WAS
1758 005236 005737 001254          TST      ERRCNT     ;IS THIS THE FIRST ERROR
1759 005242 001002          BNE      2$           ;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$:  ERROR    301        ;CONTINUE NO RESPONSE TABLE
1763 005252 012737 177777    3$:  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
```

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

```
1777
1778 005272          WRITEERROR:
1779 005272 012637 001304    MOV      (KSP)+,OLDPC  ;SAVE RETURN ADDRESS IN CASE OF ERROR LOOP
1780 005276 050037 001226    BIS      RO,ADDROR   ;LOGICAL 'OR' OF BAD ADDRESS
1781 005302 005100          COM      RO           ;GET RO READY FOR AND
1782 005304 040037 001224          BIC      RO,ADRAND   ;PERFORM LOGICAL AND
1783 005310 005100          COM      RO           ;PUT RO 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
```

```
.SBTTL SUBROUTINE TO REPORT DUAL ADDRESSING WHEN LOADING A MAP REGISTER
*****
```

1794
1795
1796

1797
1798
1799
1800
1801
1802
1803
1804
1805 005356
1806 005356 012637 001304
1807 005362 050037 001226
1808 005366 005100
1809 005370 040037 001224
1810 005374 005100
1811 005376 050137 001232
1812 005402 005101
1813 005404 040137 001230
1814 005410 005101
1815 005412 005737 001254
1816 005416 001002
1817 005420 104203
1818 005422 000401
1819 005424 104303
1820 005426 000177 173652
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832 005432 012637 001304
1833 005436 050037 001226
1834 005442 005100
1835 005444 040037 001224
1836 005450 005100
1837 005452 050337 001232
1838 005456 005103
1839 005460 040337 001230
1840 005464 005103
1841 005466 050437 001236
1842 005472 005104
1843 005474 040437 001234
1844 005500 005104
1845 005502 005737 001254
1846 005506 001002
1847 005510 104204
1848 005512 000401
1849 005514 104304
1850 005516 000177 173562
1851
1852

```

:
: THIS SUBROUTINE WILL LOG AND REPORT ALL DUAL ADDRESSING ERRORS
: FOUND IN THE MAPPING REGISTERS. A 'LOGICAL OR' AND A
: 'LOGICAL AND' OF THE WRITTEN ADDRESSES WILL BE MAINTAINED IN
: 'ADDROR' AND 'ADRAND'. THE LOG ON THE ADDITIONAL OR FAILING,
: ADDRESSES WILL BE MAINTAINED IN 'DATAOR' AND 'DATAND'.
:
:*****
DUALADR:
MOV (KSP)+,OLDPC ;STARTING LOCATION OF SUBROUTINE
BIS R0,ADDROR ;SAVE RETURN ADDRESS IN CASE OF LOOP
COM R0 ;LOGICAL OR OF WRITTEN ADDRESS
BIC R0,ADRAND ;GET R0 READY FOR AND
COM R0 ;PERFORM LOGICAL AND
BIS R1,DATAOR ;PUT R0 BACK AS IT WAS
COM R1 ;LOGICAL OR OF DUALED ADDRESS
BIC R1,DATAND ;GET R1 READY FOR AND
COM R1 ;PERFORM LOGICAL AND
TST ERRCNT ;PUT R1 BACK AS IT WAS
BNE 1$ ;SEE IF THIS IS FIRST ERROR
ERROR 203 ;BRANCH IF NOT FIRST ERROR
BR 2$ ;BRANCH TO EXIT
1$: ERROR 303
2$: JMP @OLDPC ;RETURN TO TEST

```

```

.SBTT. SUBROUTINE TO REPORT COUNT PATTERN ERRORS IN MAP REGISTERS
:*****
: THIS SUBROUTINE IS CALLED WHENEVER AN ERROR IS DISCOVERED IN
: THE COUNT PATTERN IN THE MAP REGISTERS. IT KEEPS THE LOGICAL
: 'AND' AND 'OR' OF THE FAILING REGISTER'S ADDRESS, DATA RECEIVED,
: AND PATTERN LOADED. EVERY ERROR IS REPORTED AND AT THE END OF
: A TEST AN ERROR SUMMARY IS GIVEN.
:*****
COUNT. MOV (KSP)+,OLDPC ;SAVE RETURN ADDRESS IN CASE OF ERROR LOOP
BIS R0,ADDROR ;LOGICAL OR OF BAD ADDRESSES
COM R0 ;GET R0 READY FOR AND
BIC R0,ADRAND ;PERFORM LOGICAL AND
COM R0 ;PUT R0 BACK AS IT WAS
BIS R3,DATAOR ;LOGICAL OR OF BAD DATA
COM R3 ;GET R3 READY FOR AND
BIC R3,DATAND ;PERFORM LOGICAL AND
COM R3 ;PUT R3 BACK AS IT WAS
BIS R4,PATTOR ;LOGICAL OR OF PATTERN LOADED
COM R4 ;GET R4 READY FOR AND
BIC R4,PATAND ;PERFORM LOGICAL AND
COM R4 ;PUT R4 BACK AS IT WAS
TST ERRCNT ;IS THIS FIRST ERROR
BNE 1$ ;BRANCH IF NOT FIRST ERROR
ERROR 204 ;REPORT FIRST COUNT ERROR
BR 2$ ;SKIP NEXT STATEMENT
1$: ERROR 304 ;REPORT MORE DATA
2$: JMP @OLDPC ;RETURN TO THE TEST

```

1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863 005522 012637 001304
1864 005526 050137 001236
1865 005532 005101
1866 005534 040137 001234
1867 005540 005101
1868 005542 050037 001232
1869 005546 005100
1870 005550 040037 001230
1871 005554 005100
1872 005556 005737 001254
1873 005562 001002
1874 005564 104205
1875 005566 000401
1876 005570 104305
1877 005572 000177 173506
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891 005576
1892 005576 012637 001304
1893 005602 050237 001236
1894 005606 005102
1895 005610 040237 001234
1896 005614 005102
1897 005616 050337 001232
1898 005622 005103
1899 005624 040337 001230
1900 005630 005103
1901 005632 105737 001103
1902 005636 001002
1903 005640 104207
1904 005642 000401
1905 005644 104307
1906 005646 000177 173432
1907
1908

```
.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.
*
*****
UBCOUNT:MOV      (KSP)+,OLDPC      ;SAVE RETURN ADDRESS IN CASE OF ERROR LOOP
             BIS      R1,PATTOR     ;LOGICAL OR OF COUNT LOADED
             COM      R1            ;GET R1 READY FOR AND
             BIC      R1,PATAND     ;PERFORM LOGICAL AND
             COM      R1            ;PUT R1 BACK AS IT WAS
             BIS      R0,DATAOR     ;LOGICAL OR OF DATA READ
             COM      R0            ;GET R0 READY FOR AND
             BIC      R0,DATAND     ;PERFORM LOGICAL AND
             COM      R0            ;PUT R0 BACK AS IT WAS
             TST      ERRCNT        ;IS THIS THE FIRST ERROR?
             BNE      1$           ;BRANCH IF NOT FIRST
             ERROR    205          ;REPORT FIRST ERROR ON UNIBUS DATA PATH
             BR       2$           ;SKIP NEXT INSTRUCTION
1$:          ERROR    305          ;REPORT MORE DATA FROM UNIBUS
2$:          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'.
*
*****
CACOUNT:MOV      (KSP)+,OLDPC      ;STARTING ADDRESS OF THIS SUBROUTINE
             MOV      R2,PATTOR     ;SAVE RETURN ADDRESS IN CASE OF LOOP
             BIS      R2,PATTOR     ;LOGICAL OR OF PATTERN LOADED
             COM      R2            ;GET R2 READY FOR AND
             BIC      R2,PATAND     ;PERFORM LOGICAL AND
             COM      R2            ;PUT R2 BACK AS IT WAS
             BIS      R3,DATAOR     ;LOGICAL OR OF DATA FETCHED
             COM      R3            ;GET R3 READY FOR AND
             BIC      R3,DATAND     ;PERFORM LOGICAL AND
             COM      R3            ;PUT R3 BACK AS IT WAS
             TSTB     $ERFLG        ;SEE IF THIS IS THE FIRST ERROR
             BNE      1$           ;BRANCH IF NOT FIRST ERROR
             ERROR    207          ;BRANCH TO EXIT
1$:          ERROR    307          ;RETURN TO TEST
2$:          JMP      @OLDPC
```

1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925 005652 005227
1926 005654 177777
1927 005656 001401
1928 005660 000000
1929
1930
1931
1932
1933 005662 012637 001304
1934 005666 012637 001306
1935 005672 012706 001100
1936
1937 005676 013737 177766 001266
1938 005704 013737 001304 001302
1939 005712 005737 001264
1940 005716 001406
1941 005720 023737 001266 001264
1942 005726 001403
1943 005730 104001
1944 005732 000401
1945 005734 104002
1946 005736 013737 001266 177766
1947 005744 012737 177777 005654
1948 005752 013746 001306
1949 005756 013746 001304
1950 005762 000006
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964

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

.SBTTL CPU TRAP HANDLER ROUTINE

THIS SUBROUTINE WILL HANDLE ALL CPU TRAPS AND ABORTS, THRU 'ERRVEC' (0C0004). 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

CPUER: INC (PC)+ ;MAKE FLAG ZERO IF FIRST TIME
CPFLAG: .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
10\$: MOV (KSP)+,OLDPC ;SAVE RETURN ADDRESS IN CASE OF LOOP
MOV (KSP)+,OLDPS ;SAVE OLD PSW IN CASE OF LOOP
MOV #KERSTK,KSP ;MAKE SURE THAT THE KERNEL STACK POINTER
;IS AT 1100.
MOV CPUERR,PCPUER ;SAVE CPU ERROR REGISTER
MOV OLDPC,BADPC ;SAVE PC+2 AT TIME OF ABORT
TST CPUFXP ;SEE IF ANY CONDITION WAS EXPECTED
BEQ 2\$;BRANCH IF NO TRAP WAS EXPECTED
CMP PCPUER,CPUEXP ;SEE IF EXPECTED ERROR OCCURED
BEQ 1\$;BRANCH IF ERROR CODES MATCH
ERROR 1 ;WRONG CPU TRAP CONDITION
BR 1\$;SKIP NEXT INSTRUCTION
2\$: ERROR 2 ;NO CPU TRAP EXPECTED
1\$: MOV PCPUER,CPUERR ;CLEAR CPU ERROR REGISTER
MOV #-1,CPFLAG ;MAKE FLAG NEGATIVE ONE FOR NEXT TIME
MOV OLDPS,-(KSP) ;PUSH OLD PSW BACK ON STACK
MOV OLDPC,-(KSP) ;PUSH RETURN ADDRESS BACK ON STACK
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.

```
1965 :*
1966 :*****
1967 005764 005227 MEMER: INC (PC)+ ;MAKE FLAG ZERO IF FIRST TIME
1968 005766 177777 PAFLAG: .WORD -1 ;NEGATIVE ONE FOR A FLAG
1969 005770 001401 BEQ 10$ ;BRANCH IF FIRST TIME IN
1970 005772 000000 HALT ;I HAVE ENTERED THIS ROUTINE BEFORE
1971 ;I FINISHED REPORTING THE FIRST ERROR
1972 ;THE SECOND ENTRY ADDRESS IS ON THE
1973 ;STACK AND THE FIRST ERROR CONDITION
1974 ;IS PROBABLY STILL LOCKED UP .
1975 005774 012737 006206 000114 10$: MOV #5$,CACHVEC ;SET CACHE VECTOR TO RTI UNTIL THE
1976 ;THE CACHE HAS BEEN FIXED.
1977 006002 012637 001304 MOV (KSP)+,OLDPC ;SAVE RETURN ADDRESS IN CASE OF LOOP
1978 006006 012637 001306 MOV (KSP)+,OLDPS ;SAVE OLD PSW IN CASE OF LOOP
1979 006012 013737 177740 001270 MOV @PLOADRS,PLOADR ;SAVE LOWER CACHE ADDR REG
1980 006020 013737 177742 001272 MOV @PHIADRS,PHIADR ;SAVE HIGH BITS OF FAILING ADDR
1981 006026 013737 177744 001274 MOV @MEMERR,PPARER ;SAVE MEMORY ERROR REGISTER
1982 006034 013737 177746 001276 MOV @CONTRL,PCONTR ;SAVE CONTROL REGISTER FOR TYPE OUT
1983 006042 013737 177750 001300 MOV @MAINT,PMaint ;SAVE MAINTENANCE REGISTER
1984 006050 013737 001304 001302 MOV OLDPC,BADPC ;SAVE PC+2 AT TIME OF ABORT
1985 006056 032737 000014 001274 BIT #14,PPARER ;WAS THIS A MAIN MEMORY REFERENCE
1986 006064 001005 BNE 1$ ;BRANCH IF MAIN MEMORY PARITY ERROR
1987 006066 012737 005764 000114 MOV #MEMER,CACHVEC ;LOAD ADDRESS OF THIS ROUTINE IN VECTOR
1988 006074 104003 ERROR 3 ;UNEXPECTED PARITY ABORT
1989 006076 000415 BR 2$ ;BRANCH TO SUBROUTINE EXIT
1990 006100 010046 1$: MOV RO,-(KSP) ;SAVE RO ON STACK
1991 006102 013700 001270 MOV PLOADR,RO ;PUT LOW 16 BITS OF BAD ADDR IN RO
1992 006106 042700 176000 BIC #176000,RO ;CLEAR UPPER 6 BITS OF ADDRESS
1993 006112 074027 000002 XOR RO,#BIT1 ;REFERENCE OTHER WORD OF PAIR
1994 006116 005710 TST (RO) ;READ AT SAME INDEX AS BAD WORD
1995 006120 012600 MOV (KSP)+,RO ;RESTORE RO FROM STACK
1996 006122 012737 005764 000114 MOV #MEMER,CACHVEC ;LOAD ADDRESS OF THIS ROUTINE IN VECTOR
1997 006130 104004 ERROR 4 ;MAIN MEMORY PARITY ERROR
1998 006132 005737 001322 2$: TST RETRY ;ARE YOU RETRYING THIS TEST?
1999 006136 001006 BNE 3$ ;BRANCH IF THIS IS SECOND TRY
2000 006140 005237 001322 INC RETRY ;SET RETRY FLAG
2001 006144 013737 001106 001304 MOV $LPADR,OLDPC ;RETURN TO START OF THIS TEST
2002 006152 000403 BR 4$ ;BRANCH TO EXIT
2003 006154 013737 001324 001304 3$: MOV NXTTST,OLDPC ;RETURN TO START OF NEXT TEST
2004 ;SINCE THIS IS THE SECOND ABORT
2005 006162 013737 001274 177744 4$: MOV PPARER,MEMERR ;CLEAR MEMORY ERROR REGISTER
2006 006170 012737 177777 005766 MOV #-1,PAFLAG ;RESTORE A NEGATIVE ONE FOR NEXT TIME
2007 006176 013746 001306 MOV OLDPS,-(KSP) ;PUSH OLD PSW BACK ON STACK
2008 006202 013746 001304 MOV OLDPC,-(KSP) ;PUSH RETURN ADDRESS BACK ON STACK
2009 006206 000006 5$: RTT ;RETURN FROM INTERRUPT.
```

```
2010
2011
2012
2013 .SBTTL MEMORY MANAGEMENT TRAPS AND ABORTS HANDLER ROUTINE
2014 :*****
2015 :*
2016 :* THIS ROUTINE WILL HANDLE ALL SPURIOUS MEMORY MANAGEMENT TRAPS
2017 :* AND ABORTS. IT WILL REPORT THE CONDITION OF ALL THE MEMORY
2018 :* MANAGEMENT STATUS REGISTERS, AND THEN RETURN TO THE TEST AND
2019 :* TRY TO CONTINUE RUNNING.
2020 :*
```



```
2021  
2022 006210 005227  
2023 006212 177777  
2024 006214 001401  
2025 006216 000000  
2026  
2027  
2028  
2029  
2030 006220 011637 001302  
2031 006224 012637 001304  
2032 006230 012637 001306  
2033 006234 013737 177572 001312  
2034 006242 013737 177574 001314  
2035 006250 013737 177576 001316  
2036 006256 104005  
2037 006260 042737 177776 177572 1$:  
2038 006266 012737 177777 006212  
2039 006274 013746 001306  
2040 006300 013746 001304  
2041 006304 000006  
2042  
2043  
2044  
2045  
2046  
2047  
2048 010000  
2049  
2050 010000  
2051 010000 012737 000340 177776  
2052 010006 012706 001100  
2053 010012 005026  
2054 010014 022706 001136  
2055 010020 001374  
2056 010022 012706 001100  
2057 010026 012737 002214 000020  
2058 010034 012737 000340 000022  
2059 010042 012737 002534 000030  
2060 010050 012737 000340 000032  
2061 010056 012737 004506 000034  
2062 010064 012737 000340 000036  
2063 010072 012737 004550 000024  
2064 010100 012737 000340 000026  
2065 010106 013737 024426 024420  
2066 010114 005037 001204  
2067 010120 005037 001206  
2068 010124 112737 000001 001115  
2069 010132 012737 024646 000014  
2070 010140 012737 000340 000016  
2071 010146 012737 000002 024646  
2072 010154 012737 010202 000010  
2073 010162 005046  
2074 010164 012746 010172  
2075 010170 000006  
2076 010172 012737 000006 024646 64$:
```

```
*****  
MMTRAP: INC (PC)+ ;MAKE FLAG ZERO IF FIRST TIME  
MMFLAG: .WORD -1 ;FLAG SHOULD BE NEG ONE  
BEQ 10$ ;BRANCH IF FIRST TIME INTO ROUTINE  
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  
10$: MOV (KSP),BADPC ;SAVE PC AT TIME OF ABORT OR TRAP  
MOV (KSP)+,OLDPC ;SAVE RETURN ADDRESS IN CASE OF LOOP  
MOV (KSP)+,OLDPS ;SAVE OLD PSW IN CASE OF LOOP  
MOV MMR0,PMR0 ;SAVE STATUS REGISTER  
MOV MMR1,PMR1 ;SAVE AUTO I/C/DEC REGISTER  
MOV MMR2,PMR2 ;SAVE VIRTUAL ADDRESS REGISTER  
ERROR 5 ;UNEXPECTED M.M. ABORT OR TRAP  
1$: BIC #177776,@MMR0 ;CLEAR ALL BITS EXCEPT 0  
MOV #-1,MMFLAG ;RESTORE A NEGATIVE ONE TO FLAG  
MOV OLDPS,-(KSP) ;PUSH OLD PSW ONTO STACK  
MOV OLDPC,-(KSP) ;PUSH RETURN ADDRESS ON STACK  
RTT ;RETURN TO MAIN PROGRAM  
*****  
.SBTTL *****  
.SBTTL ***** START OF TEST CODE *****  
.=10000 ;START TEST CODE AT ADDRESS 10000 (2K)  
*****  
START:  
MOV #340,@PPS ;:LOCK OUT ALL INTERRUPTS  
MOV #SCMTAG,R6 ;:FIRST LOCATION TO BE CLEARED  
CLR (R6)+ ;:CLEAR MEMORY LOCATION  
CMP #STKS,R6 ;:DONE?  
BNE .-6 ;:LOOP BACK IF NO  
MOV #STACK,SP ;:SETUP THE STACK POINTER  
MOV #SCOPE,@IOTVEC ;:IOT VECTOR FOR SCOPE ROUTINE  
MOV #340,@IOTVEC+2 ;:LEVEL 7  
MOV #ERROR,@EMTVEC ;:EMT VECTOR FOR ERROR ROUTINE  
MOV #340,@EMTVEC+2 ;:LEVEL 7  
MOV #STRAP,@TRAPVEC ;:TRAP VECTOR FOR TRAP CALLS  
MOV #340,@TRAPVEC+2 ;:LEVEL 7  
MOV #SPURDN,@PURVEC ;:POWER FAILURE VECTOR  
MOV #340,@PURVEC+2 ;:LEVEL 7  
MOV SENDCT,SEOPCT ;:SETUP END-OF-PROGRAM COUNTER  
CLR $TIMES ;:INITIALIZE NUMBER OF ITERATIONS  
CLR $ESCAPE ;:CLEAR THE ESCAPE ON ERROR ADDRESS  
MOVB #1,$ERMAX ;:ALLOW ONE ERROR PER TEST  
MOV #SRTN,@TBITVEC ;:SET 'T' BIT VECTOR TO SRTN  
MOV #340,@TBITVEC+2 ;:LEVEL 7  
MOV #RTI,$SRTN ;:SET SRTN TO A RTI  
MOV #65$,@RESVEC ;:TRY TO DO A RTT  
CLR -(SP) ;:DUMMY PS  
MOV #64$,-(SP) ;:AND PC  
RTT ;:TRY THE RTT  
MOV #RTT,$SRTN ;:RTT IS LEGAL--SET SRTN 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    #SENDAD,#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$: .ASCIZ <CRLF>?CEKBFDP 11/70 UNIBUS MAP?<CRLF>
2090 010314                                     67$:
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: CLR    @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
2108                                     ;;KB11CM (11/74)
2109 010334 012737 000001 001330  MOV    #1,@KB11E    ;;HERE IF KB11E OR KB11EM. SET FLAG
2110 010342 005037 177750          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 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 DMA 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 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\$: TST	@KB11E	:IS IS A KB11-E OR KB11-EM?	
2148	010536	005737	001330		BEQ	1\$:BR IF NEITHER. MUST BE KB11CM	
2149	010542	001404			MOV	#BIT8,@KB11E	:SET UPPER BYTE (KB11-EM)	
2150	010544	012737	000400	001330	BR	2\$:DONE	
2151	010552	000402			1\$: INCB	@KB11CM	:YES, FLAG THIS AS A MODIFIED PROCESSOR	
2152	010554	105237	001332		2\$: BR	ENDKB	:DONE DETERMINING WHICH CPU	
2153	010560	000403						
2154					MFPTR: MOV	#T1,(SP)	:HERE IF MFPTR TRAPPED. SEE IF 1170 OR KB11CM	
2155	010562				RTI		:SET UP RETURN ADDRESS FOR RTI	
2156	010562	012716	010342		ENDKB: INC	#-1	:RETURN	
2157	010566	000002			BNE	100\$:FIRST TIME?	
2158	010570				TYPE	,MSG1	:BR IF NO	
2159	010570	005227	177777		TST	@KB11E	:<15><12>CPU UNDER TEST FOUND TO BE A	
2160	010574	001026			BNE	101\$:IS THIS A KB11-E OR KB11-EM?	
2161	010576	104400	024662		TSTB	@KB11CM	:BR IF EITHER ONE	
2162	010602	005737	001330		BNE	1\$:IS IT A 11/74 (KB11CM)	
2163	010606	001011			TYPE	,MSG3	:BR IF IT IS	
2164	010610	105737	001332		BR	100\$:KB11-B/C<15><12>	
2165	010614	001003			1\$: TYPE	,MSG4	:SKIP OTHER MESSAGE	
2166	010616	104400	024732		BR	100\$:11/74 (KB11CM)<15><12>	
2167	010622	000413			101\$: TSTB	@KB11E	:SKIP CISP MESSAGE	
2168	010624	104400	024744		BEQ	102\$:IS IT A KB11-E?	
2169	010630	000410			TYPE	,MSG5	:BR IF NOT, MUST BE KB11-EM	
2170	010632	105737	001330		BR	100\$:KB11-E<15><12>	
2171	010636	001403			102\$: TYPE	,MSG2	:SKIP KB11-EM MESSAGE	
2172	010640	104400	024775		100\$: LOOP: MOV	#MEMER,CACHVEC	:KB11-EM<15><12>	
2173	010644	000402					:LOAD PARITY ERROR SERVICE	
2174	010646	104400	024721				:ROUTINE ADDRESS	
2175	010652						:SET PRIORITY SEVEN	
2176	010652	012737	005764	000114			:LOAD MEMORY MANAGEMENT TRAP	
2177							:SERVICE ROUTINE ADDRESS	
2178	010660	012737	000340	000116			:SET PRIORITY SEVEN	
2179	010666	012737	006210	000250			:LOAD CPU TRAP SERVICE ROUTINE ADDR	
2180							:SET PRIORITY SEVEN	
2181	010674	012737	000340	000252			:LOAD ROM ADDRESS OF 'NOP' INTO MICRO	
2182	010702	012737	005652	000004			:BREAK REGISTER. EVERY 'NOP' WILL	
2183	010710	012737	000340	000006			:GENERATE A 'SYNC' PULSE ON PIN	
2184	010716	012727	000044	177770			:#AE1 SLOT 10	
2185							:NOT EXPECTING ANY CPU ERRORS	
2186								
2187								
2188	010724	005037	001264		CLR	CPUEXP		

2189	010730	005037	177572
2190	010734	005037	172516
2191	010740	012700	177777
2192	010744	010037	005654
2193	010750	010037	005154
2194	010754	010037	005766
2195	010760	010037	006212
2196	010764	010037	177744
2197	010770	010037	177766

```

CLR @MMR0 ;START IN 16 BIT MAPPING
CLR @MMR3 ;DISABLE MAP AND 22-BIT MAPPING
MOV #-1,R0 ;NEGATIVE ONE USED TO INITIALIZE FLAGS
MOV R0,CPFLAG ;INITIALIZE FLAGS
MOV R0,TOFLAG ;INITIALIZE FLAGS
MOV R0,PAFLAG ;INITIALIZE FLAGS
MOV R0,MMFLAG ;INITIALIZE FLAGS
MOV R0,@MEMERR ;CLEAR PARITY ERROR REGISTER
MOV R0,@CPUERR ;CLEAR CPU ERROR REGISTER

```

```

: *THE FOLLOWING TWO TESTS ARE USED TO VERIFY THAT SOMETHING
: *RESPONDS WHEN THE MAP REGISTERS AND CACHE REGISTERS ARE
: *REFERENCED. THE SIGNALS THAT SHOULD BE GENERATED ARE:
: *'MAPB REG OP H' AND 'MAPB CACHE REG H', UNDER PROPER
: *OPERATION EITHER SIGNAL SHOULD CAUSE 'BUS SSYN L' TO BE
: *ASSERTED AND NO TIMEOUT WILL OCCUR.

```

```

:*****
: *TEST 1 MAP REGISTER RESPONSE TEST

```

```

: * THIS TEST IS USED TO ENSURE THAT ALL THE UNIBUS MAP REGISTERS
: * CAN BE REFERENCED UNDER PROGRAM CONTROL, WITHOUT TIMING OUT.
: * THE ADDRESSES OF ANY MAP REGISTERS THAT TIME OUT WILL BE REPORTED
: * AND, AT THE END OF THE TEST, A SUMMARY OF THOSE REGISTERS WILL
: * BE GIVEN. THE ADDRESS DECODE CIRCUITRY IS ON 'MAPB' AND 'MAPC'.

```

```

:*****
: TST1:

```

2216	010774			
2217	010774	000004		
2218	010776	012737	011060	001324
2219				
2220	011004	012737	005152	000004
2221	011012	012700	170200	
2222	011016	012737	011024	001110
2223	011024	000240		
2224	011026	011002		
2225	011030	062700	000002	
2226	011034	022700	170376	
2227	011040	103371		
2228	011042	012737	011004	001110
2229				
2230	011050	005737	001254	
2231	011054	001401		
2232	011056	104006		

```

MOV #TST2,NXTTST ;SAVE STARTING ADDRESS OF NEXT TEST
;FOR ESCAPE ON PARITY ERRORS
20$: MOV #TIMEOUT,ERRVEC ;LOAD ERRVEC WITH ROUTINE ADDR.
MOV #MAPLO,R0 ;PUT FIRST MAP REG ADDR IN R0
MOV #1$,SLPERR ;SET LOOP ON ERROR POINTER TO 1$
1$: NOP ;THIS IS A SYNC POINT FOR SCOPING
MOV (R0),R2 ;READ MAP REGISTERS TO R2
ADD #2,R0 ;POINT TO NEXT MAP REGISTER
CMP #MAPH37,R0 ;COMPARE LAST MAP REG ADDR WITH R0
BHS 1$ ;CONTINUE READING IF NOT AT LAST REG.
MOV #20$,SLPERR ;INITIALIZE LOOPING POINTER IN CASE
;OF INTERMITTENT FAULTS.
TST ERRCNT ;SEE IF THERE WERE ANY ERRORS
BEQ TST2 ;NO ERRORS GO TO NEXT TEST
ERROR 6 ;SUMMARY OF MAP REGISTERS THAT TIMED OUT

```

```

:*****
: *TEST 2 CACHE REGISTER RESPONSE TEST

```

```

: * THIS TEST IS USED TO ENSURE THAT ALL THE CACHE REGISTERS
: * CAN BE REFERENCED UNDER PROGRAM CONTROL, WITHOUT TIMING OUT.
: * THE ADDRESSES OF ANY CACHE REGISTERS THAT TIME OUT WILL BE
: * REPORTED AND, AT THE END OF THE TEST, A SUMMARY OF THOSE REGISTERS
: * WILL BE GIVEN.
: * THESE REGISTERS ARE TESTED HERE BECAUSE THE ADDRESS DECODE AND DATA
: * PATH IS ON THE UNIBUS MAP BOARD (8141). THE ADDRESS DECODE CIRCUITRY

```

2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244

2245
2246
2247
2248 011060
2249 011060 000004
2250 011062 012737 011136 001324
2251
2252 011070 012700 177740
2253 011074 012737 011102 001110
2254 011102 000240
2255 011104 011002
2256 011106 062700 000002
2257 011112 022700 177752
2258 011116 103371
2259 011120 012737 011070 001110
2260
2261 011126 005737 001254
2262 011132 001401
2263 011134 104007
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286 011174
2287 011136 000004
2288 011140 012737 011234 001324
2289
2290 011146 012737 005652 000004
2291 011154 012700 170200
2292 011160 012737 011166 001110
2293 011166 000240
2294 011170 005010
2295 011172 011037 001170
2296 011176 001402
2297 011200 004737 005272
2298 011204 062700 000004
2299 011210 022700 170274
2300 011214 103364

IS ON 'MAPB' AND GENERATES 'MAPB CACHE REG'.

TST2:
SCOPE
MOV #TST3,NXTTST ;SAVE STARTING ADDRESS OF NEXT TEST
;FOR ESCAPE ON PARITY ERRORS
20\$: MOV #LOADRS,R0 ;PUT ADDR OF FIRST CACHE REG IN R0
MOV #1\$,SLPERR ;SET LOOP ON ERROR POINTER HERE
1\$: NOP ;THIS IS A SYNC POINT FOR SCOPING
MOV (R0),R2 ;TRY TO READ ALL CACHE REGISTERS
ADD #2,R0 ;POINT TO NEXT CACHE REGISTER
CMP #HITMIS,R0 ;HAVE ALL REGISTERS BEEN REFERENCED?
BHS 1\$;BRANCH IF MORE TO TEST
MOV #20\$,SLPERR ;INITIALIZE LOOPING ERROR IN CASE
;OF INTERMITTENT FAULTS
TST ERRCNT ;WERE THERE ANY ERRORS.
BEQ TST3 ;BRANCH IF NO ERRORS
ERROR 7 ;SUMMARY OF CACHE REGISTERS THAT TIMED OUT

*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
*ZEROS. 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.

TST3:
SCOPE
MOV #TST4,NXTTST ;SAVE STARTING ADDRESS OF NEXT TEST
;FOR ESCAPE ON PARITY ERRORS
20\$: MOV #CPUER,ERRVEC ;LOAD CPU TRAP SERVICE ROUTINE ADDR
MOV #MAPL0,R0 ;LOAD STARTING MAP REGISTER ADDR
MOV #1\$,SLPERR ;SET LOOP ON ERROR POINTER HERE
1\$: NOP ;THIS IS A SYNC POINT FOR SCOPING
CLR (R0) ;CLEAR MAP REGISTER
MOV (R0),\$TMP0 ;SEE IF MAP REGISTER IS ZERO
BEQ 2\$;BRANCH IF REGISTER IS ZERO
JSR PC,WRITEERROR ;REPORT AND LOG ERROR
2\$: ADD #4,R0 ;POINT TO NEXT MAP REG LOWER 16 BITS
CMP #MAPL17,R0 ;SEE IF THIS SECTION IS TESTED
BHS 1\$;BRANCH IF NOT

```
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
```

*TEST 4 CLEAR AND READ LOW 20 REGISTERS UPPER 6 BITS

THIS TEST WILL ENSURE THAT MAPH00 - MAPH17 WILL ALL HOLD ZERO.
IF ANY REGISTERS ARE NOT ZERO AFTER BEING CLEARED, THEIR ADDRESS AND
CONTENTS ARE REPORTED. IF ALL THE REGISTERS HAVE RANDOM DATA THEN
PROBABLY NO WRITE PULSE IS BEING GENERATED, BUT IF THE ERROR IS IN
ONLY ONE REGISTER OR IN ONE BIT IT MIGHT BE MORE DIFFICULT TO FIND.
IT IS POSSIBLE THAT THE COUNT PATTERN TEST FOR MAPH00 - MAPH17 WILL
GIVE MORE HELP.

*TST4:

```
2321 011234           SCOPE
2322 011234 000004    MOV    #TST5,NXTTST   ;SAVE STARTING ADDRESS OF NEXT TEST
2323 011236 012737 011324 001324    MOV    #20$, $LPERR   ;FOR ESCAPE ON PARITY ERRORS
2324                                     ;LOAD STARTING MAP REGISTER ADDR
2325 011244 012700 170202 001110 20$: MOV    #MAPH0,RO      ;SET LOOP ON ERROR POINTER HERE
2326 011250 012737 011256 001110 1$:  NOP                    ;THIS IS A SYNC POINT FOR SCOPING
2327 011256 000240    CLR    (RO)           ;CLEAR MAP REGISTER
2328 011260 005010    MOV    (RO), $TMP0    ;SEE IF MAP REGISTER IS ZERO
2329 011262 011037 001170    BEQ    2$             ;BRANCH IF REGISTER IS ZERO
2330 011266 001402    JSR    PC,WRITEERROR ;REPORT AND LOG ERROR
2331 011270 004737 005272 2$:  ADD    #4,RO          ;POINT TO NEXT MAP REG UPPER 6 BITS
2332 011274 062700 000004    CMP    #MAPH17,RO     ;SEE IF TESTING IS OVER
2333 011300 022700 170276    BHIS  1$             ;BRANCH IF MORE REGS TO TEST
2334 011304 103364    MOV    #20$, $LPERR   ;INITIALIZE LOOPING POINTER IN CASE
2335 011306 012737 011244 001110    MOV    #20$, $LPERR   ;OF INTERMITTENT FAULTS.
2336                                     ;WERE THERE ANY ERRORS ON THIS
2337 011314 005737 001254    TST    ERRCNT          ;BRANCH IF NO ERRORS
2338 011320 001401           BEQ    TST5            ;SUMMARY OF MAP REGS THAT COULD NOT
2339 011322 104011           ERROR   11            ;HOLD ZERO
```

*TEST 5 CLEAR AND READ HIGH 20 REGISTERS LOWER 16 BITS

THIS TEST WILL ENSURE THAT MAPL20 - MAPL37 WILL ALL HOLD ZERO.
IF ANY REGISTERS ARE NOT ZERO AFTER BEING CLEARED, THEIR ADDRESS AND
CONTENTS ARE REPORTED. IF ALL THE REGISTERS HAVE RANDOM DATA THEN
PROBABLY NO WRITE PULSE IS BEING GENERATED, BUT IF THE ERROR IS IN
ONLY ONE REGISTER OR IN ONE BIT IT MIGHT BE MORE DIFFICULT TO FIND.
IT IS POSSIBLE THAT THE COUNT PATTERN TEST FOR MAPL20 - MAPL37 WILL
GIVE MORE HELP.

*TST5:

```
2355 011324           SCOPE
2356 011324 000004
```

```

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$:    MOV      #MAPL20,R0      ;LOAD STARTING MAP REGISTER ADDR
2360 011340 012737 011346 001110        MOV      #1$,SLPERR        ;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$,SLPERR      ;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
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388

```

```

*****
*TEST 6          CLEAR AND READ HIGH 20 REGISTERS UPPER 6 BITS
*****

```

```

THIS TEST WILL ENSURE THAT MAPH20 - MAPH37 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 MAPH20 - MAPH37 WILL
GIVE MORE HELP.

```

```

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$:    MOV      #MAPH20,R0      ;LOAD STARTING MAP REGISTER ADDR
2394 011430 012737 011436 001110        MOV      #1$,SLPERR        ;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$,SLPERR      ;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
2411
2412

```

```

;*IN THE NEXT TWO TESTS A COUNT PATTERN IS RUN THROUGH
;*MAP REGISTER 00 TO TEST THE DATA PATH TO AND FROM THE
;*MAP REGISTERS. THE DATA RECEIVERS ARE ON PAGE MAPA, AND

```

2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468

: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'.

TST7:

```
SCOPE
MOV #TST10,NXTTST ;SAVE STARTING ADDRESS OF NEXT TEST
;FOR ESCAPE ON PARITY ERRORS
;DO 12 ITERATIONS
20$: MOV #12,$TIMES ;START COUNT PATTERN AT 0
CLR R3 ;START COUNT PATTERN AT 0
CLR R2 ;SET LOOP ON ERROR POINTER TO 1$
1$: MOV #1$,SLPERR ;THIS IS A SYNC POINT FOR SCOPING
NOP ;LOAD MAP REG 0 LOW 16 BITS
MOV R2,MAPL00 ;READ CONTENTS OF MAP REGISTER 0
MOV MAPL00,R0 ;COMPARE DATA & PATTERN
CMP R2,R0 ;BRANCH IF DATA DOES NOT MATCH
BEQ 2$ ;POSSIBLE DATA PATH FAILURE
ERROR 12 ;BRANCH TO TEST DATA PATH WITH
BR 3$ ;A REGISTER OF THE NEXT GROUP
2$: ADD #2,R2 ;ADD 2 TO COUNT PATTERN
BNE 1$ ;BRANCH IF COUNT NOT COMPLETE.
BR 10$ ;DATA PATH OKAY FOR LOW 16 BITS
3$: MOV #12$,SLPERR ;SET LOOP ON ERROR POINTER TO 12$
12$: NOP ;THIS IS A SYNC POINT FOR SCOPING
MOV R3,MAPL20 ;LOAD MAP REG 20 LOW 16 BITS
MOV MAPL20,R1 ;READ DATA STORED IN MAP REGISTER 20
CMP R3,R1 ;COMPARE DATA & PATTERN
BNE 4$ ;CAUGHT ERROR,NOW SEE IF SAME.
ADD #2,R3 ;ADD 2 TO COUNT PATTERN
BNE 2$ ;BRANCH IF COUNT NOT 0
4$: ERROR 13 ;PROBABLY IS A REAL DATA PATH ERROR
10$: MOV #20$,SLPERR ;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

2469
2470
2471
2472
2473
2474
2475
2476 011630
2477 011630 000004
2478 011632 012737 012030 001324
2479
2480 011640 012737 000002 001204
2481 011646 005002 20\$:
2482 011650 005003
2483 011652 005004
2484 011654 105737 001331
2485 011660 001003
2486 011662 105737 001332
2487 011666 001403
2488 011670 012737 012026 012022 55\$:
2489 011676 012737 011704 001110 56\$:
2490 011704 000240 1\$:
2491 011706 010402
2492 011710 010237 170202
2493 011714 013700 170202
2494 011720 047702 000076
2495 011724 020200
2496 011726 001402
2497 011730 104012
2498 011732 000405
2499 011734 005204 2\$:
2500 011736 022704 177777
2501 011742 001360
2502 011744 000422
2503 011746 012737 011754 001110 3\$:
2504 011754 000240 12\$:
2505 011756 010403
2506 011760 010337 170302
2507 011764 013701 170302
2508 011770 047703 000026
2509 011774 020301
2510 011776 001004
2511 012000 005204
2512 012002 022704 177777
2513 012006 001352
2514 012010 104013 4\$:
2515 012012 012737 011646 001110 10\$:
2516 012020 000403
2517 012022 012024 72\$:
2518 012024 177700
2519 012026 077700
2520
2521
2522
2523
2524

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'.

TST10:
SCOPE
MOV #TST11,NXTTST ;SAVE STARTING ADDRESS OF NEXT TEST
;FOR ESCAPE ON PARITY ERRORS
MOV #2,\$TIMES ;DO 2 ITERATIONS
CLR R2 ;START COUNT PATTERN AT 0
CLR R3 ;START COUNT PATTERN AT 0
CLR R4 ;START COUNT AT ZERO
TSTB KB11EM ;IS THIS A KB11-EM?
BNE 55\$;BR IF IT IS
TSTB KB11CM ;IS THIS A MODIFIED PROCESSOR?
BEQ 56\$;NO,CARRY ON
MOV #72\$+4,72\$;USE THE MODIFIED MASK
MOV #1\$,SLPERR ;SET LOOP ON ERROR POINTER TO 1\$
NOP ;THIS IS A SYNC POINT FOR SCOPING
MOV R4,R2 ;MOVE COUNT TO R2
MOV R2,@MAPH0 ;LOAD MAP REG 00 HIGH 6 BITS
MOV MAPH0,R0 ;READ BACK MAP REG 00 UPPER SIX BITS
BIC @72\$,R2 ;MASK OUT UNUSED BITS
CMP R2,R0 ;COMPARE DATA AND PATTERN LOADED
BEQ 2\$;BRANCH IF DATA IS CORRECT
ERROR 12 ;POSSIBLE DATA PATH ERROR
BR 3\$;BRANCH TO TRY SECOND BANK OF MAP REGS
INC R4 ;CHANGE DATA PATTERN
CMP #177777,R4 ;HAS COUNT REACH MAXIMUM?
BNE 1\$;BRANCH IF TEST NOT OVER
BR 10\$;UPPER SIX BIT DATA PATH IS OKAY
MOV #12\$,SLPERR ;SET LOOP ON ERROR POINTER TO 12\$
NOP ;THIS IS A SYNC POINT FOR SCOPING
MOV R4,R3
MOV R3,@MAPH20 ;LOAD MAP REG 20 HIGH 6 BITS
MOV MAPH20,R1 ;READ BACK MAP REG 20 UPPER SIX BITS
BIC @72\$,R3
CMP R3,R1 ;COMPARE DATA AND PATTERN
BNE 4\$;BRANCH IF DATA DOES NOT MATCH
INC R4 ;CHANGE DATA PATTERN
CMP #177777,R4 ;HAS COUNT REACHED MAXIMUM?
BNE 2\$;BRANCH IF TEST NOT OVER
ERROR 13 ;PROBABLY IS A REAL DATA PATH ERROR
MOV #20\$,SLPERR ;SET LOOP POINTER TO START OF TEST
BR 72\$+6
;POINTER TO PROPER MASK
.WORD 72\$+2 ;NONMODIFIED PROCESSORS USE THIS
.WORD 177700 ;MODIFIED PROCESSORS USE THIS
.WORD 077700

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

2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580

DURING A 'MOV #DATA, @MAPREG' INSTRUCTION, ALL MAP REGISTERS
ARE CLEARED AND ONE REGISTER AT A TIME, STARTING WITH MAPLOO,
IS LOADED A NEGATIVE ONE. THEN, ALL MAP REGISTERS ARE READ,
STARTING WITH MAPH37, AND VERIFIED TO BE ZERO. ANY REGISTER
THAT IS NOT ZERO AND WHOSE UNIBUS ADDRESS DOES NOT MATCH THAT
OF THE REGISTER UNDER TEST IS REPORTED TO BE IN ERROR. AT THE
END OF THE TEST A SUMMARY OF ALL DUALED REGISTERS IS GIVEN.
THE SIGNALS THAT ARE TESTED HERE ARE ON 'MAPC' AND ARE:
'EN LOREG' & 'EN HIREG'. THE 'A' INPUTS TO THE ADDRESS
MULTIPLEXER SHOULD BE THE ONES IN USE DURING A READ OR WRITE
TO THE UNIBUS MAP REGISTERS.

```
TST11:
SCOPE
MOV #TST12,NXTTST ;SAVE STARTING ADDRESS OF NEXT TEST
;FOR ESCAPE ON PARITY ERRORS
;DO 144 ITERATIONS
MOV #144,$TIMES
MOV #MAPLO,R0 ;LOAD FIRST MAP REG ADDR INTO R0
MOV #1$, $LPERR ;SET LOOP ON ERROR POINTER HERE
MOV #MAPH37,R1 ;PUT ADDRESS OF HIGHEST MAP REG IN R1
JSR PC,CLRMAP ;CLEAR ALL MAP REGISTERS IN CASE OF
;A 'FLAKEY' PROBLEM
;THIS IS A SYNC POINT FOR SCOPING
MOV #177777,(R0) ;LOAD MAP REGISTER UNDER TEST
MOV (R1),R2 ;READ MAP REGS STARTING FROM TOP
BEQ 3$ ;GO TO 3$ IF ZERO
CMP R1,R0 ;SEE IF NON-ZERO ADDRS MATCH
BEQ 3$ ;GO TO 3$ IF MATCH
JSR PC,DUALADR ;JUMP TO SUBROUTINE TO LOG ALL ERRORS
SUB #2,R1 ;POINT TO NEXT MAP REGISTER
CMP #MAPLO,R1 ;HAVE ALL REGISTERS BEEN READ
BLOS 2$ ;IF NOT CONTINUE TEST
ADD #2,R0 ;POINT TO NEXT MAP REGISTER
CMP #MAPH37,R0 ;SEE IF TEST IS OVER
BHS 1$ ;CONTINUE TESTING
MOV K20$, $LPERR ;INITIALIZE LOOPING POINTER IN CASE
;OF INTERMITTENT FAULTS.
;SEE IF THERE WERE ANY ERRORS
TST ERRCNT ;GOTO NEXT TEST IF NO ERRORS
BEQ TST12 ;SUMMARY OF DUAL ADDRESSING ERRORS
ERROR 14 ;ON LOADING MAP REGISTERS
```

:THE NEXT FOUR(4) TESTS RUN A COUNT PATTERN THROUGH EACH
:MAP REGISTER, CHECKING FOR STUCK BITS, AND SHORTED PINS.

:TEST 12 COUNT PATTERN LOW 20 MAP REGISTERS LOWER 16 BITS

THIS TEST WILL RUN A COUNT PATTERN THROUGH UNIBUS MAP REGISTERS
MAPLOO - MAPL17. IF THE COUNT RECEIVED 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'.

```

2581
2582
2583
2584 012154
2585 012154 000004
2586 012156 012737 012300 001324
2587
2588 012164 012737 000012 001204
2589 012172 005002
2590 012174 012737 012214 001110
2591 012202 010204
2592 012204 042704 000001
2593 012210 012700 170200
2594 012214 000240
2595 012216 010210
2596 012220 011003
2597 012222 020403
2598 012224 001402
2599 012226 004737 005432
2600 012232 005003
2601 012234 062700 000004
2602 012240 022700 170274
2603 012244 103363
2604 012246 022702 177777
2605 012252 001403
2606 012254 062702 000401
2607 012260 000750
2608 012262 012737 012172 001110
2609
2610 012270 005737 001254
2611 012274 001401
2612 012276 104000

```

```

TST12:
SCOPE
MOV #TST13,NXTTST ;SAVE STARTING ADDRESS OF NEXT TEST
;FOR ESCAPE ON PARITY ERRORS
;DO 12 ITERATIONS
MOV #12,$TIMES
CLR R2 ;START WITH AN ALL ZERO PATTERN
MOV #2,$SLPERR ;SET LOOP ON ERROR POINTER TO 2$
MOV R2,R4 ;PUT COUNT IN R4 FOR MASK
BIC #000001,R4 ;CLEAR BITS IN MASK FOR PROPER COMPARE
MOV #MAPL00,R0 ;LOAD STARTING MAP REGISTER ADDRESS
;THIS IS A SYNC POINT FOR SCOPING
NOP
MOV R2,(R0) ;LOAD MAP REGISTER WITH COUNT PATTERN
MOV (R0),R3 ;READ MAP REGISTER INTO R3
CMP R4,R3 ;COMPARE MASK WITH DATA
BEQ 3$ ;BRANCH IF DATA MATCHES
JSR PC,COUNT ;JUMP TO COUNT TO LOG ALL ERRORS
CLR R3 ;CLEAR DATA HOLDER
ADD #4,R0 ;POINT TO NEXT MAP REGISTER UNDER TEST
CMP #MAPL17,R0 ;SEE IF ALL REGISTERS HAVE BEEN LOADED
BHS 2$ ;BRANCH IF STILL MORE TO GO
CMP #177777,R2 ;SEE IF COUNT HAS CYCLED
BEQ 4$ ;BRANCH IF TEST IS DONE
ADD #401,R2 ;CHANGE COUNT PATTERN
BR 1$ ;CONTINUE TESTING
MOV #20$,$SLPERR ;INITIALIZE LOOPING POINTER IN CASE
;OF INTERMITTENT FAULTS.
TST ERRCNT ;SEE IF THERE WERE ANY ERRORS
BEQ TST13 ;BRANCH IF NO ERRORS
ERROR ;SUMMARY OF COUNT PATTERN FAILURES
;IN MAPPING REGISTERS

```

```

2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628 012300
2629 012300 000004
2630 012302 012737 012424 001324
2631
2632 012310 012737 000144 001204
2633 012316 005002
2634 012320 012737 012340 001110
2635 012326 010204
2636 012330 042704 177700

```

```

*****
TEST 13 COUNT PATTERN LOW 20 MAP REGISTERS UPPER 6 BITS
*****
THIS TEST WILL RUN A COUNT PATTERN THROUGH UNIBUS MAP REGISTERS
MAP00 - MAP17. IF THE COUNT RECEIVED DOES NOT MATCH THE
EXPECTED PATTERN THEN THE MAP REGISTER ADDRESS, DATA RECEIVED,
PATTERN LOADED, AND PATTERN EXPECTED ARE REPORTED. AT THE END
OF THE TEST A SUMMARY OF ALL ERRORS IS GIVEN SO THAT YOU CAN
DETERMINE IF THE ERROR IS BIT SENSITIVE OR REGISTER SENSITIVE.
THE REGISTERS ARE ON 'MAPC' AND THE OUTPUT MULTIPLEXER IS ON
'MAPJ'.
*****

```

```

TST13:
SCOPE
MOV #TST14,NXTTST ;SAVE STARTING ADDRESS OF NEXT TEST
;FOR ESCAPE ON PARITY ERRORS
;DO 144 ITERATIONS
MOV #144,$TIMES
CLR R2 ;START WITH AN ALL ZERO PATTERN
MOV #2,$SLPERR ;SET LOOP ON ERROR POINTER TO 2$
MOV R2,R4 ;PUT COUNT IN R4 FOR MASK
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      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    R3,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      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
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672

```
*****
*TEST 14      COUNT PATTERN HIGH 20 MAP REGISTERS LOWER 16 BITS
*
* THIS TEST WILL RUN A COUNT PATTERN THROUGH UNIBUS MAP REGISTERS
* MAPL20 - MAPL37. IF THE COUNT RECEIVED 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'.
*****
```

```
2673 012424 000004 012550 001324  TST14: SCOPE
2674 012424 000004      MOV    #TST15,NXTTST  ;SAVE STARTING ADDRESS OF NEXT TEST
2675 012426 012737 012550 001324      MOV    #12,$TIMES     ;FOR ESCAPE ON PARITY ERRORS
2676                                ;DO 12 ITERATIONS
2677 012434 012737 000012 001204 20$: MOV    #2$,$LPERR     ;START WITH AN ALL ZERO PATTERN
2678 012442 005002      CLR    R2              ;SET LOOP ON ERROR POINTER TO 2$
2679 012444 012737 012464 001110 1$: MOV    R2,R4           ;PUT COUNT IN R4 FOR MASK
2680 012452 010204      BIC    #000001,R4     ;CLEAR BITS IN MASK FOR PROPER COMPARE
2681 012454 042704 000001      MOV    #MAPL20,R0     ;LOAD STARTING MAP REGISTER ADDRESS
2682 012460 012700 170300 2$: NOP                    ;THIS IS A SYNC POINT FOR SCOPING
2683 012464 000240      MOV    R2,(R0)         ;LOAD MAP REGISTER WITH COUNT PATTERN
2684 012466 010210      MOV    (R0),R3         ;READ MAP REGISTER INTO R3
2685 012470 011003      CMP    R4,R3           ;COMPARE MASK WITH DATA
2686 012472 020403      BEQ    3$              ;BRANCH IF DATA MATCHES
2687 012474 001402      JSR    PC,COUNT        ;JUMP TO COUNT TO LOG ALL ERRORS
2688 012476 004737 005432 3$: CLR    R3              ;CLEAR DATA HOLDER
2689 012502 005003      ADD    #4,R0           ;POINT TO NEXT MAP REGISTER UNDER TEST
2690 012504 062700 000004      CMP    #MAPL37,R0     ;SEE IF ALL REGISTERS HAVE BEEN LOADED
2691 012510 022700 170374      BHIS   2$              ;BRANCH IF STILL MORE TO GO
2692 012514 103363
```

2693	012516	022702	177777			CMP	#177777,R2	:SEE IF COUNT HAS CYCLED
2694	012522	001403				BEQ	4\$:BRANCH IF TEST IS DONE
2695	012524	062702	000401			ADD	#401,R2	:CHANGE COUNT PATTERN
2696	012530	000750				BR	1\$:CONTINUE TESTING
2697	012532	012737	012442	001110	4\$:	MOV	#20\$,\$LPERR	:INITIALIZE LOOPING POINTER IN CASE
2698								:OF INTERMITTENT FAULTS.
2699	012540	005737	001254			TST	ERRCNT	:SEE IF THERE WERE ANY ERRORS
2700	012544	001401				BEQ	TST15	:BRANCH IF NO ERRORS
2701	012546	104015				ERROR	15	:SUMMARY OF COUNT PATTERN FAILURES
2702								:IN MAPPING REGISTERS

2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748

: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 RECEIVED 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'.

TST15:

2717	012550					SCOPE		
2718	012550	000004				MOV	#TST16,NXTTST	:SAVE STARTING ADDRESS OF NEXT TEST
2719	012552	012737	012674	001324				:FOR ESCAPE ON PARITY ERRORS
2720								:DO 144 ITERATIONS
2721	012560	012737	000144	001204		MOV	#144,\$TIMES	:START WITH AN ALL ZERO PATTERN
2722	012566	005002			20\$:	CLR	R2	:SET LOOP ON ERROR POINTER TO 2\$
2723	012570	012737	012610	001110		MOV	#2\$,\$LPERR	:PUT COUNT IN R4 FOR MASK
2724	012576	010204			1\$:	MOV	R2,R4	:CLEAR BITS IN MASK FOR PROPER COMPARE
2725	012600	042704	177700			BIC	#177700,R4	:LOAD STARTING MAP REGISTER ADDRESS
2726	012604	012700	170302			MOV	#MAPH20,R0	:THIS IS A SYNC POINT FOR SCOPING
2727	012610	000240			2\$:	NOP		:LOAD MAP REGISTER WITH COUNT PATTERN
2728	012612	010210				MOV	R2,(R0)	:READ MAP REGISTER INTO R3
2729	012614	011003				MOV	(R0),R3	:COMPARE MASK WITH DATA
2730	012616	020403				CMP	R4,R3	:BRANCH IF DATA MATCHES
2731	012620	001402				BEQ	3\$:JUMP TO COUNT TO LOG ALL ERRORS
2732	012622	004737	005432			JSR	PC,COUNT	:CLEAR DATA HOLDER
2733	012626	005003			3\$:	CLR	R3	:POINT TO NEXT MAP REGISTER UNDER TEST
2734	012630	062700	000004			ADD	#4,R0	:SEE IF ALL REGISTERS HAVE BEEN LOADED
2735	012634	022700	170376			CMP	#MAPH37,R0	:BRANCH IF STILL MORE TO GO
2736	012640	103363				BHIS	2\$:SEE IF COUNT HAS CYCLED
2737	012642	022702	000177			CMP	#177,R2	:BRANCH IF TEST IS DONE
2738	012646	001403				BEQ	4\$:CHANGE COUNT PATTERN
2739	012650	062702	000001			ADD	#1,R2	:CONTINUE TESTING
2740	012654	000750				BR	1\$:INITIALIZE LOOPING POINTER IN CASE
2741	012656	012737	012566	001110	4\$:	MOV	#20\$,\$LPERR	:OF INTERMITTENT FAULTS.
2742								:SEE IF THERE WERE ANY ERRORS
2743	012664	005737	001254			TST	ERRCNT	:BRANCH IF NO ERRORS
2744	012670	001401				BEQ	TST16	:SUMMARY OF COUNT PATTERN FAILURES
2745	012672	104016				ERROR	16	:IN MAPPING REGISTERS

2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767 012674
2768 012674 000004
2769 012676 012737 013072 001324
2770
2771 012704 012737 012724 001110 20\$:
2772 012712 005000
2773 012714 032737 000014 177746
2774 012722 001012
2775 012724 000240 10\$:
2776 012726 005005
2777 012730 005037 177746
2778 012734 013700 177746 21\$:
2779 012740 001403
2780 012742 005205
2781 012744 001373
2782 012746 104017
2783 012750 012737 012756 001110 1\$:
2784 012756 000240 11\$:
2785 012760 005037 177750
2786 012764 013701 177750
2787 012770 001401
2788 012772 104020
2789 012774 012737 013002 001110 2\$:
2790 013002 012737 177777 177744 12\$:
2791 013010 000240
2792 013012 013700 177740
2793 013016 022700 177740
2794 013022 001401
2795 013024 104021
2796 013026 012737 013034 001110 3\$:
2797 013034 012737 177777 177744 13\$:
2798 013042 000240
2799 013044 013700 177742
2800 013050 042700 177700
2801 013054 022700 000003
2802 013060 001401
2803 013062 104022
2804 013064 012737 012704 001110 14\$:

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 MAINTENANCE 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.

TST16:
SCOPE
MOV #TST17,NXTTST ;SAVE STARTING ADDRESS OF NEXT TEST
;FOR ESCAPE ON PARITY ERRORS
MOV #10\$,\$LPERR ;SET LOOP ON ERROR POINTER TO 10\$
CLR R0 ;CLEAR R0 IN CASE CACHE IS DISABLED
BIT #14,@CONTRL ;SEE IF EITHER GROUP IS DISABLED
BNE 1\$;BRANCH IF CACHE IS DISABLED
NOP ;THIS IS A SYNC POINT FOR SCOPING
CLR R5
CLR @CONTRL ;CLEAR CONTROL REGISTER
MOV @CONTRL,R0 ;READ CONTROL REGISTER
BEQ 1\$;BRANCH IF CONTROL REGISTER IS ZERO
INC R5 ;WAIT FOR VCIP BIT IN CCR TO
BNE 21\$;CLEAR
ERROR 17 ;POSSIBLE FAULT IN CACHE REG DATA PATHS
MOV #11\$,\$LPERR ;SET LOOP ON ERROR POINTER TO 11\$
NOP ;THIS IS A SYNC POINT FOR SCOPING
CLR @MAINT ;CLEAR MAINTENANCE REGISTER
MOV @MAINT,R1 ;READ MAINTENANCE REGISTER
BEQ 2\$;BRANCH IF MAINTENANCE REGISTER IS ZERO
ERROR 20 ;MAINTENANCE REGISTER WILL NOT CLEAR
MOV #12\$,\$LPERR ;SET LOOP ON ERROR POINTER TO 12\$
MOV #-1,MEMERR ;MAKE SURE THAT CACHE ERROR REG IS CLEAR
NOP ;THIS IS A SYNC POINT FOR SCOPING
MOV LOADRS,R0 ;READ CACHE LOW ADDR REG TO R0
CMP #177740,R0 ;SEE IF R0 = ADDR OF CACHE LOW ADDR REG
BEQ 3\$;BRANCH IF DATA READ CORRECTLY
ERROR 21 ;CANNOT READ 177740 FROM 'LOADRS'
MOV #13\$,\$LPERR ;SET LOOP ON ERROR POINTER TO 13\$
MOV #-1,MEMERR ;MAKE SURE THAT CACHE ERROR REG IS CLEAR
NOP ;THIS IS A SYNC POINT FOR SCOPING
MOV HIADRS,R0 ;READ CACHE HIGH ADDR REG TO R0
BIC #177700,R0 ;MASK OFF UPPER 10 BITS
CMP #3,R0 ;SEE IF R0 = 3
BEQ 14\$;BRANCH IF DATA MATCHES
ERROR 22 ;CANNOT READ 000003 FROM 'HIADRS'
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
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860

*TEST 17 CACHE REGISTER DATA PATH COUNT PATTERN
*
* THIS TEST FIRST SAVES THE CONDITION OF THE CACHE CONTROL REGISTER IS '\$TMP5' SO THAT THE CACHE CAN BE RESTORED. IF EITHER BIT 3 OR 4 IS ON THE TEST BRANCHES TO '\$\$' TO TEST BITS 5 - 15 OF THE DATA PATH, IF NEITHER BIT IS ON THEN THE TEST RUNS A COUNT PATTERN THRU THE CONTROL REGISTER FROM 00 - 77. IF ANY BITS ARE DETECTED AS BEING STUCK THE ERROR IS REPORTED AND A SUMMARY OF THOSE ERRORS IS GIVEN. THEN THE TEST FORCES MISSES IN BOTH GROUPS OF THE CACHE AND PROCEEDS TO RUN A COUNT PATTERN THROUGH THE CACHE MAINTENANCE REGISTER (177746) FROM 000000 - 177760 IN INCREMENTS OF 20. THE CODE IS SET UP IN SUCH A WAY THAT NO PARITY ERRORS SHOULD BE GENERATED. AGAIN ALL BAD MATCHES OF DATA WILL BE REPORTED AND A SUMMARY OF ALL ERRORS IS GIVEN AT THE END OF THE TEST AND THE CACHE IS RESTORED TO ITS PREVIOUS CONDITION.

TST17:
 SCOPE
 MOV #TST20,NXTTST ;SAVE STARTING ADDRESS OF NEXT TEST
 ;FOR ESCAPE ON PARITY ERRORS
 MOV #12,\$TIMES ;DO 12 ITERATIONS
 TBIT0 ;TURN OFF T BIT TRAPPING IF ON
 MOV @#CONTRL,\$TMP5 ;SAVE CONTROL REG TO RESTORE CACHE
 BIT #14,@#CONTRL ;SEE IF EITHER GROUP IS DISABLED
 BNE \$\$;SKIP FIRST PART OF TEST IF DISABLED
 MOV #CONTRL,R0 ;LOAD ADDR OF CONTROL REG INTO R0
 CLR R2 ;START COUNT PATTERN AT ZERO
 MOV #1\$,SLPERR ;SET LOOP ON ERROR POINTER TO 1\$
 1\$: NOP ;THIS IS A SYNC POINT FOR SCOPING
 MOV R2,(R0) ;LOAD COUNT INTO CONTROL REG
 MOV (R0),R3 ;READ COUNT FROM CONTROL REG
 CMP R2,R3 ;SEE IF COUNT = DATA READ
 BEQ \$\$;BRANCH IF DATA MATCHES
 JSR PC,CACOUNT ;LOG AND REPORT COUNT PATTERN ERROR
 2\$: INC R2 ;CHANGE COUNT PATTERN
 CMP #77,R2 ;SEE IF COUNT HAS PASSED MAXIMUM
 BHS 1\$;BRANCH IF COUNT HASN'T CYCLED
 TST ERRCNT ;SEE IF ANY ERRORS ON THIS CYCLE
 BEQ \$\$;BRANCH IF NO ERRORS
 ERROR 23 ;COUNT THRU CONTROL REG FAILED
 CLR RETRY ;CLEAR RETRY FLAG AN THE START OF
 ;EACH TEST
 CLR ERRCNT ;CLEAR THE MULTIPLE ERROR COUNTER
 CLR DATAOR ;LOCATION FOR LOGICAL OR OF BAD DATA
 CLR ADDROR ;LOCATION FOR LOGICAL OR OF ADDRESS
 CLR PATTOR ;LOCATION FOR LOGICAL OR OF PATTERN LOADED
 MOV #-1,R0 ;LOAD -1 INTO R0 TO INITIALIZE LOGICAL AND LOCS
 MOV R0,DATAND ;LOCATION FOR LOGICAL AND OF BAD DATA
 MOV R0,ADRAND ;LOCATION FOR LOGICAL AND OF ADDRESS
 MOV R0,PATAND ;LOCATION FOR LOGICAL AND OF PATTERN LOADED
 3\$: MOV #14,@#CONTRL ;DISABLE BOTH GROUPS OF CACHE

2861	013254	012700	177750			MOV	#MAINT,R0	:LOAD ADDR OF MAINTENANCE 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\$,SLPERR	: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\$,SLPERR	: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						TBITR		:RESTORE THE T BIT TO ITS CONDITION
2888	013352	104420						:BEFORE THE LAST TEST
2889						MOV	#77406,R0	:MAKE THE KERNEL I-SPACE PAGES ALL
2890	013354	012700	077406					:4K, UPWARD EXPANDABLE, READ/WRITE
2891						MOV	R0,KIPDR0	:KERNEL I-SPACE PAGE 0
2892	013360	010037	172300			MOV	R0,KIPDR1	:KERNEL I-SPACE PAGE 1
2893	013364	010037	172302			MOV	R0,KIPDR2	:KERNEL I-SPACE PAGE 2
2894	013370	010037	172304			MOV	R0,KIPDR3	:KERNEL I-SPACE PAGE 3
2895	013374	010037	172306			MOV	R0,KIPDR4	:KERNEL I-SPACE PAGE 4
2896	013400	010037	172310			MOV	R0,KIPDR5	:KERNEL I-SPACE PAGE 5
2897	013404	010037	172312			MOV	R0,KIPDR6	:KERNEL I-SPACE PAGE 6
2898	013410	010037	172314			MOV	R0,KIPDR7	:KERNEL I-SPACE PAGE 7
2899	013414	010037	172316			MOV	#000,KIPAR0	:MAP TO PHYSICAL 0
2900	013420	012737	000000	172340		MOV	#200,KIPAR1	:MAP TO PHYSICAL 4K - 8K
2901	013426	012737	000200	172342		MOV	#400,KIPAR2	:MAP TO PHYSICAL 8K - 12K
2902	013434	012737	000400	172344		MOV	#600,KIPAR3	:MAP TO PHYSICAL 12K - 16K
2903	013442	012737	000600	172346		MOV	#1000,KIPAR4	:MAP TO PHYSICAL 16K - 20K
2904	013450	012737	001000	172350		MOV	#1200,KIPAR5	:MAP TO PHYSICAL 20K - 24K
2905	013456	012737	001200	172352		MOV	#170000,KIPAR6	:MAP TO UNIBUS
2906	013464	012737	170000	172354		MOV	#177600,KIPAR7	:MAP TO I/O PAGE
2907	013472	012737	177600	172356		MOV	#BIT0,MMR0	:ENABLE FULL 18-BIT MAPPING
2908	013500	012737	000001	177572		MOV	#BIT4,MMR3	:ENABLE 22-BIT MAPPING
2909	013506	012737	000020	172516				

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

2917
2918
2919
2920
2921
2922 013514
2923 013514 000004
2924 013516 012737 013546 001106
2925 013524 012737 013546 001110
2926 013532 012737 000020 001102
2927 013540 013737 001102 177570
2928
2929 013546 012737 000020 001264 20\$:
2930 013554 012737 013614 001110
2931 013562 012737 177702 172354
2932 013570 012702 175254
2933 013574 010237 170200
2934 013600 012700 004000
2935 013604 012704 140000
2936 013610 074037 172354 1\$:
2937 013614 000240 10\$:
2938 013616 011401
2939 013620 020201
2940 013622 001007
2941 013624 012714 000000
2942 013630 005737 170200
2943 013634 001001
2944 013636 104025
2945
2946 013640 010114 2\$:
2947 013642 074037 172354 3\$:
2948 013646 006200
2949 013650 020027 000001
2950 013654 001355
2951 013656 012737 013546 001110
2952 013664 005037 001264

TO SELECT MAPLOO, THRU KIPAR6. THE TEST THEN CHANGES ONE BIT AT A TIME IN PAR6 SO THAT IT SHOULD NEVER REFERENCE MAPLOO. IF IT DOES AN ERROR IS REPORTED.

```
*****
TST20:
SCOPE
MOV #20$,SLPADR ;SET LOOP ON TEST POINTER TO 20$
MOV #20$,SLPERR ;SET LOOP ON ERROR POINTER TO 20$
MOV #20$,STSTM ;SETUP TEST NUMBER AND CLR ERROR FLAG
MOV $STSTM,DISPLAY ;DISPLAY TEST NUMBER FOR ALL TO SEE
.EQUIV BIT4,TIMOUT ;BIT4 IS TIMEOUT BIT IN CPU ERROR REG
MOV #TIMOUT,CPUEXP ;EXPECTING CPU TIME OUT ON UNIBUS
20$: MOV #10$,SLPERR ;SET LOOP ON ERROR POINTER TO 10$
MOV #177702,KIPAR6 ;PUT MAP REG 0 ADDR IN PAR6
MOV #175254,R2 ;PATTERN FOR TESTING.
MOV R2,@MAPLO ;LOAD MAP REGISTER 0
MOV #BIT11,R0 ;SET BIT 11 TO FLOAT THRU PAR6
MOV #140000,R4 ;VIRT.ADDR. TO SELECT PAR6
1$: XOR R0,KIPAR6 ;CHANGE A BIT OF MAP REG 0'S ADDR
10$: NOP ;THIS IS A SYNC POINT FOR SCOPING
MOV (R4),R1 ;READ LOCATION POINTED TO BY PAR6
CMP R2,R1 ;SEE IF DATA FETCHED MATCHES PATTERN
BNE 3$ ;BRANCH IF NOT SAME
MOV #0,(R4) ;TRY TO CLEAR THIS LOCATION
TST @MAPLO ;SEE IF MAP REG 0 GOT CLEARED
BNE 2$ ;BRANCH IF MAP REG NOT ZERO
ERROR 2$ ;GOT TO MAPLO WITH ONE BIT
;DIFFERENT IN ADDRESS FROM 770200
2$: MOV R1,(R4) ;RELOAD THE LOCATION
3$: XOR R0,KIPAR6 ;RESTORE BIT TO ORIGINAL STATUS
ASR R0 ;RIGHT SHIFT ONE PLACE
CMP R0,#1 ;SEE IF TEST IS OVER
BNE 1$ ;BRANCH TO CONTINUE TEST
MOV #20$,SLPERR ;SET LOOP POINTER TO START OF TEST
CLR CPUEXP ;ZERO EXPECTED CPU TRAP FLAG
*****
```

*TEST 21 CACHE REGISTER ADDRESS DECODE TEST

THIS TEST TRIES TO VERIFY THAT NONE OF THE INPUTS TO THE ADDRESS DECODER FOR THE CACHE REGISTERS ON 'MAPB' IS STUCK TRUE. KIPAR6 IS SETUP TO HOLD 177740 AND R4 HAS THE VIRTUAL ADDRESS TO SELECT 'LOADRS', THRU KIPAR6. THE TEST THEN CHANGES ONE BIT AT A TIME IN PAR6 SO THAT IT SHOULD NEVER REFERENCE 'LOADRS'. IF IT DOES AN ERROR IS REPORTED.

2953
2954
2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965 013670
2966 013670 000004
2967 013672 012737 014072 001324
2968
2969 013700 012737 000020 001264 20\$:
2970 013706 012737 013742 001110
2971 013714 012737 177777 172354
2972 013722 012702 177740

```
*****
TST21:
SCOPE
MOV #TST22,NXTTST ;SAVE STARTING ADDRESS OF NEXT TEST
;FOR ESCAPE ON PARITY ERRORS
20$: MOV #TIMOUT,CPUEXP ;EXPECTING UNIBUS TIMEOUT IN THIS TEST
MOV #10$,SLPERR ;SET LOOP ON ERROR POINTER TO 10$
MOV #177777,KIPAR6 ;PUT CACHE REG BASE IN PAR6
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\$,SLPERR	: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	3\$:	MOV	#12\$,SLPERR	: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	4\$:	MOV	#13\$,SLPERR	: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 ADRS 177740
3005	014056	104027				ERROR	27	:READ CACHE 'LOADRS' WITHOUT ADRS 777740
3006	014060	012737	013700	001110	19\$:	MOV	#20\$,SLPERR	:SET LOOP POINTER TO START OF TEST
3007	014066	005037	001264			CLR	(PUEXP	:ZERO EXPECTED CPU TRAP CONDITION

3008
3009
3010
3011
3012
3013
3014
3015
3016
3017
3018
3019
3020
3021
3022
3023
3024
3025
3026
3027 014072
3028 014072 000004

```

*****
*TEST 22 DATA PATH, UNIBUS TO MAIN MEMORY
*
* THIS TEST RUNS A COUNT PATTERN THROUGH A MEMORY LOCATION VIA
* THE UNIBUS. THE UNIBUS MAP IS LEFT OFF DURING THIS TEST SO
* THAT THE ADDRESS IS NOT RELOCATED. THE TEST TRIES TO LOAD THE
* PATTERN INTO ADDRESS 040000 (8K) BUT IF THE MAP JUMPERS ARE
* SET NOT TO RESPOND TO THAT ADDRESS THE NEXT 4K IS TRIED UNTIL
* THE TEST GETS TO MAIN MEMORY FROM THE UNIBUS. IF THIS TEST
* DETERMINES THAT IT CANNOT GET TO MAIN MEMORY FROM THE UNIBUS
* IT REPORTS THE FACT AND SKIPS THE NEXT TEST FOR VERIFICATION.
* THE DATA PATH TO MAIN MEMORY FROM THE UNIBUS IS ON 'MAPA' AND
* THE PATH FROM MAIN MEMORY TO THE UNIBUS IS FROM THE CACHE
* 'DTML CDMX D00' - 'DTML CDMX D15' INTO 'MAPH' THE '2' INPUTS
* TO THE MULTIPLEXER AND THEN TO 'MAPJ' AND OUT TO THE UNIBUS.
*****
TST22: 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$,SLPERR       ;SET LOOP ON ERROR POINTER TO 10$
3035 014130 012737 170400 172354      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      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      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$:         CLR      R1               ;CLEAR REGISTER TO HOLD COUNT
3056 014246 012737 014260 001110      MOV      #11$,SLPERR       ;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$:      MOV      #20$,SLPERR       ;SET LOOP POINTER TO START OF TEST
3071 014320 005037 001264      CLR      CPUEXP           ;ZERO EXPECTED CPU TRAP CONDITION

```

```

3072
3073
3074
3075
3076
3077
3078
3079
3080
3081
3082
3083
3084

```

```

*****
*TEST 23      MAP DOESN'T RELOCATE IF NOT ENABLED
*
*      THIS TEST VERIFIES THAT THE UNIBUS MAP DOES NOT RELOCATE IF BITS
*      OF MMR3 IS NOT SET. THE TEST ASSUMES THAT THE PREVIOUS TEST HAS
*      RUN SUCCESSFULLY AND LEFT KIPAR6 POINTING TO THE FIRST UNIBUS
*      MAPPING REGISTER THAT THE UNIBUS MAP WILL RESPOND TO GREATER
*      THAN OR EQUAL TO MAPREG #2. KIPAR5 IS ALSO POINTING TO THE
*      SAME MEMORY BASE ADDRESS EXCEPT IT POINTS OVER THE FASTBUS.
*      THE TEST THEN SETS ONE BIT IN EACH A.L.U. OF THE UNIBUS MAP
*      AND TRIES TO REFERENCE MAIN MEMORY OVER THE UNIBUS. SINCE THE

```

MAP IS NOT ENABLED THE LOAD WILL GO TO MAIN MEMORY UNRELOCATED.

```

3085
3086
3087
3088 014324
3089 014324 000004
3090 014326 012737 014436 001324
3091
3092 014334 012737 000020 001264 20$
3093 014342 012737 014400 001110
3094 014350 013700 172354
3095 014354 072027 177773
3096 014360 042700 007400
3097 014364 012720 021042
3098 014370 012710 000042
3099 014374 005037 120000
3100 014400 000240
3101 014402 012737 043207 140000
3102
3103
3104
3105 014410 013703 120000
3106 014414 022703 043207
3107 014420 001401
3108 014422 104032
3109 014424 012737 014334 001110 1$
3110 014432 005037 001264
3111
3112
3113
3114
3115
3116
3117
3118
3119
3120
3121
3122
3123
3124
3125
3126 014436
3127 014436 000004
3128 014440 012737 015074 001324
3129
3130 014446 012737 000001 001204
3131 014454
3132 014454 000004
3133 014456 012737 014506 001106
3134 014464 012737 014506 001110
3135 014472 012737 000024 001102
3136 014500 013737 001102 177570
3137 014506 012737 000020 001264 20$
3138 014514 012700 170200
3139 014520 012720 020000 1$
3140 014524 005020

```

TST23:

```

SCOPE
MOV #TST24,NXTTST ;SAVE STARTING ADDRESS OF NEXT TEST
;FOR ESCAPE ON PARITY ERRORS
;MIGHT TIME OUT OVER UNIBUS
MOV #10$,SLPERR ;SET LOOP ON ERROR POINTER TO 10$
MOV KIPAR6,R0 ;PUT UNIBUS ADDRESS OF MAP REG IN R0
ASH #5,R0 ;RIGHT SHIFT R0 5 PLACES
BIC #007400,R0 ;STRIP OFF EXTRANEIOUS BITS.
MOV #021042,(R0)+ ;SET BOTTOM BIT IN EACH ALU
MOV #42,(R0) ;SET BOTTOM BIT IN EACH ALU
CLR #120000 ;CLEAR TEST LOCATION VIA FAST BUS
10$: NOP ;THIS IS A SYNC POINT FOR SCOPING
MOV #43207,#140000 ;LOAD TEST LOCATION VIA UNIBUS
;THIS LOAD SHOULD NOT BE RELOCATED
;BY THE UNIBUS MAP, SINCE BIT05 OF
;MMR3 IS CLEAR.
MOV #120000,R3 ;READ TEST LOCATION VIA FAST BUS
CMP #43207,R3 ;SEE IF DATA MATCHES
BEQ 1$ ;BRANCH IF DATA GOOD
ERROR 32 ;MAP RELOCATED WHEN NOT ENABLED
MOV #20$,SLPERR ;SET LOOP POINTER TO START OF TEST
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'.

TST24:

```

SCOPE
MOV #TST25,NXTTST ;SAVE STARTING ADDRESS OF NEXT TEST
;FOR ESCAPE ON PARITY ERRORS
;DO 1 ITERATION
MOV #1,$TIMES
SIZEJ:
SCOPE
MOV #20$,SLPADR ;SET LOOP ON TEST POINTER TO 20$
MOV #20$,SLPERR ;SET LOOP ON ERROR POINTER TO 20$
MOV #24,$STSTM ;SETUP TEST NUMBER AND CLR ERROR FLAG
MOV $STSTM,DISPLAY ;DISPLAY TEST NUMBER FOR ALL TO SEE
MOV #TIMOUT,CPUEXP ;EXPECTING CPU TIME OUT ON UNIBUS
20$: MOV #MAPLO,R0 ;LOAD ADDRESS OF FIRST MAP REG
MOV #20000,(R0)+ ;LOAD 4K INTO ALL MAP REGISTERS
1$: 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	#BITS,@MMR3	:TURN ON MAP RELOCATION
3144	014542	012790	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 TRIFD
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					:::			
3190					:::			
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
 3198 015034 062701 170200
 3199 015040 062700 170200
 3200 015044 010137 001244
 3201 015050 062701 000002
 3202 015054 010137 001246
 3203 015060 010037 001250
 3204 015064 062700 000002
 3205 015070 010037 001252

ASH #5,R1 ;RIGHT SHIFT R1 5 PLACES
 ADD #170200,R1
 ADD #170200,R0
 MOV R1,LREGL
 ADD #2,R1 ;POINT TO UPPER BITS OF MAP REG
 MOV R1,LREGU
 MOV R0,HREGL
 ADD #2,R0 ;POINT TO UPPER BITS OF MAP REG
 MOV R0,HREGU

3206
 3207
 3208
 3209
 3210
 3211
 3212
 3213
 3214
 3215
 3216
 3217
 3218
 3219
 3220
 3221
 3222

 :TEST 25 ENSURE THAT THERE IS NO DUAL MAPPING

THIS TEST VERIFIES THAT THE OTHER INPUT (X) TO THE ADDRESS
 MULTIPLEXER ON 'MAPC' IS FUNCTIONING PROPERLY. IT CLEARS
 ALL THE MAP REGISTERS EXCEPT THE ONE UNDER TEST, AND LOADS
 THAT ONE WITH 00020000. THE TEST THEN USES A VIRTUAL ADDRESS
 TO SELECT THAT MAP REGISTER AND ADD 17776, SO THAT IT SHOULD
 REFERENCE ADDRESS 00037776. A REFERENCE IS MADE THROUGH EACH
 OF THE REGISTERS AND ANY THAT FETCH THE CORRECT DATA ARE CHECKED
 TO SEE THAT IT WAS THE MAP REGISTER UNDER TEST. IF NOT BOTH THE
 MAP REGISTER UNDER TEST AND THE DUALED REGISTER ARE REPORTED.

3223 015074
 3224 015074 000004
 3225 015076 012737 015270 001324
 3226
 3227 015104 012737 000144 001204
 3228 015112 004737 005134 20\$:
 3229 015116 005037 001170
 3230 015122 012703 100000
 3231 015126 012737 015160 001110
 3232 015134 013702 001244
 3233 015140 013700 001240
 3234 015144 013701 001240 1\$:
 3235 015150 012712 040000
 3236 015154 010237 040000
 3237
 3238 015160 010137 172350 2\$:
 3239 015164 011304
 3240 015166 020402
 3241 015170 001010
 3242 015172 020001
 3243 015174 001403
 3244 015176 004737 005356
 3245 015202 000403
 3246 015204 012737 000001 001170 3\$:
 3247 015212 062701 000200 4\$:
 3248 015216 023701 001242
 3249 015222 103356
 3250 015224 005737 001170
 3251 015230 001001
 3252 015232 104036

 TST25:

SCOPE
 MOV #TST26,NXTTST ;SAVE STARTING ADDRESS OF NEXT TEST
 ;FOR ESCAPE ON PARITY ERRORS
 ;DO 144 ITERATIONS
 MOV #144,\$TIMES
 JSR PC,CLRMAP ;CLEAR ALL MAP REGISTERS
 CLR \$TMP0 ;USED AS FLAG IN TEST
 MOV #100000,R3 ;SELECT P.A.R. 4 OFFSET OF ZERO
 MOV #2\$,\$LPERR ;SET LOOP ON ERROR POINTER TO 2\$
 MOV LREGL,R2 ;PUT ADDRESS OF LOWEST USABLE MAP REG IN R2
 MOV LOWEST,R0 ;MAP REGISTER UNDER TEST IN R0
 MOV LOWEST,R1 ;MAP REGISTER USED IN CURRENT REFERENCE
 MOV #40000,(R2) ;LOAD MAP REG UNDER TEST WITH BK BASE
 MOV R2,#40000 ;LOAD TEST LOCATION WITH THE ADDRESS
 ;OF THE MAP REGISTER UNDER TEST
 MOV R1,KIPAR4 ;LOAD PAR 4 WITH NEXT MAP REG UNIBUS ADDR
 MOV (R3),R4 ;READ THROUGH THE MAP
 CMP R4,R2 ;SEE IF CORRECT DATA WAS FETCHED
 BNE 4\$;BRANCH IF NO MATCH
 CMP R0,R1 ;SEE IF MAP REGS ARE THE SAME
 BEQ 3\$;BRANCH IF CORRECT MAP REG WAS USED
 JSR PC,DUALADR ;LOG AND REPORT ALL ERRORS
 BR 4\$;SKIP NEXT INSTRUCTION
 MOV #1,\$TMP0 ;SET FLAG WHEN ADDR MATCH
 ADD #200,R1 ;TRY NEXT MAP REG
 CMP HIGEST,R1 ;SEE IF ALL HAVE BEEN TRIED
 BHIS 2\$;BRANCH IF STILL MORE TO TRY
 TST \$TMP0 ;SEE THAT THERE WAS A SUCCESSFUL MATCH
 BNE 5\$;BRANCH IF THERE WAS
 ERROR 36 ;DID NOT MATCH MAP REG ADDRESS

```

3253 015234 005037 001170
3254 015240 005012
3255 015242 062702 000004
3256 015246 062700 000200
3257 015252 023700 001242
3258 015256 103332
3259 015260 005737 001254
3260 015264 001401
3261 015266 104013
3262
3263
3264
3265
3266
3267
3268
3269
3270
3271
3272
3273
3274
3275 015270
3276 015270 000004
3277 015272 012737 015550 001324
3278
3279 015300 042737 000040 172516
3280 015306 012737 000400 172350
3281 015314 012700 040000
3282 015320 012701 100000
3283 015324 012702 010000
3284 015330 010021
3285 015332 062700 000002
3286 015336 077204
3287 015340 062737 000200 172350
3288 015346 022737 001400 172350
3289 015354 101361
3290
3291
3292
3293 015356 012737 015364 001106
3294 015364 012737 015422 001110
3295 015372 022737 171400 172354
3296
3297 015400 101463
3298 015402 013700 172354
3299
3300 015406 072027 000006
3301 015412 012701 140000
3302 015416 012702 010000
3303 015422 000240
3304 015424 011103
3305 015426 020003
3306 015430 001015
3307 015432 062701 000002
3308 015436 062700 000002

```

```

5$: CLR $TMP0 ;CLEAR FLAG FOR NEXT REG
CLR (R2) ;CLEAR MAP REG JUST TESTED
ADD #4,R2 ;POINT TO NEXT MAP REG TO LOAD
ADD #200,R0 ;POINT TO NEXT MAP REG UNDER TEST
CMP HIGEST,R0 ;SEE IF ALL MAP REGS HAVE BEEN TESTED
BHS 1$ ;BRANCH IF STILL MORE TO TEST
TST ERRCNT ;SEE IF THERE WERE ANY ERRORS
BEQ TST26 ;BRANCH TO NEXT TEST IF NO ERRORS
ERROR 13 ;ERROR TYPE OUT ITEM 13

```

```

*****
*TEST 26 LOAD LOCATIONS 40000 - 137776 WITH THEIR ADDRESSES
*
* THIS TEST IS USED TO LOAD MAIN MEMORY FROM ADDRESS 00040000 TO
* ADDRESS 00137776 WITH ITS OWN ADDRESS. IT THEN CHECKS THAT
* MEMORY OVER THE UNIBUS AND LOGS AND REPORTS ANY ERRORS THAT
* IT FINDS.
*****

```

```

TST26:
SCOPE
MOV #TST27,NXTTST ;SAVE STARTING ADDRESS OF NEXT TEST
;FOR ESCAPE ON PARITY ERRORS
BIC #BIT5,MMR3 ;TURN OFF MAP RELOCATION
MOV #400,KIPAR4 ;MAP PAGE 4 TO 8K
MOV #40000,R0 ;STARTING ADDRESS FOR DATA PATTERN
1$: MOV #100000,R1 ;VIRTUAL ADDRESS
MOV #*D4096,R2 ;LOAD 4096 LOCATIONS AT A TIME
2$: MOV R0,(R1)+ ;LOAD PHY. ADDR. INTO EACH MEMORY LOC.
ADD #2,R0 ;POINT TO NEXT PHYSICAL ADDRESS
SOB R2,2$ ;BRANCH IF 4K OF MEMORY NOT LOADED
ADD #200,KIPAR4 ;POINT TO NEXT 4K BANK OF MEMORY
CMP #1400,KIPAR4 ;SEE IF 24K IS LOADED
BHI 1$ ;BRANCH IF MORE MEMORY TO LOAD

*
* MEMORY FROM 8K - 24K IS NOW LOADED WITH ITS OWN ADDRESS
*
MOV #20$,SLPADR ;SET LOOP ADDRESS TO 20$
20$: MOV #4$,SLPERR ;SET LOOP ON ERROR POINTER TO 4$
CMP #171400,KIPAR6 ;DID I USE ANY MAP REGISTER
;BELOW REGISTER 6 (UB. ADDR 140000)
BLOS TST27 ;BRANCH TO NEXT TEST IF NOT
MOV KIPAR6,R0 ;LOAD PAR6 INTO R0 TO GET
;THE STARTING DATA PATTERN
;R0 NOW HOLDS THE STARTING DATA PATTERN
3$: MOV #6,R0 ;STARTING VIRTUAL ADDRESS
MOV #*D4096,R2 ;PREPARE TO READ 4K AT A TIME
4$: NOP ;THIS IS A SYNC POINT FOR SCOPING
MOV (R1),R3 ;READ MAIN MEMORY THRU UNIBUS
CMP R0,R3 ;SEE IF THE ADDRESSES MATCH
BNE 6$ ;BRANCH IF ERROR
5$: ADD #2,R1 ;CHANGE VIRTUAL ADDRESS
ADD #2,R0 ;CHANGE PHYSICAL ADDRESS

```



```
3365                                     ;MAP REGISTER (DEFAULT MAP REG. 0).
3366                                     ;PHYSICAL ADDRESS 17700000 IS THEN
3367                                     ;GENERATED, WHICH SHOULD TIME OUT SINCE
3368                                     ;IT IS THE FIRST NON-EXISTANT LOCATION.
3369 015652 022737 000020 001266      CMP      #TIMOUT,PCPUER ;THE UNIBUS SHOULD HAVE TIMED OUT
3370 015660 001401                    BEQ      10$             ;BRANCH IF CONDITION WAS CORRECT
3371 015662 104040                    ERROR   40             ;UNIBUS DID NOT TIME OUT
3372 015664 012737 015574 001110 10$: MOV      #20$,SLPERR   ;SET LOOP POINTER TO START OF TEST
3373 015672 005037 001264                    CLR      CPUEXP       ;NO CPU TRAPS EXPECTED FOR AWHILE
3374
3375
3376 *****
3377 *TEST 30      RELOCATION TEST USING LOWEST USABLE MAPPING REG
3378
3379      THIS TEST CHECKS OUT THE FULL ADDITION PROPERTIES OF THE UNIBUS
3380      MAP A.L.U..  IN THE DEFAULT CASE IT USES MAP REGISTER ZERO BUT
3381      IF THE MAP JUMPERS HAVE BEEN ALTERED TO DE-SELECT SOME MAP REGISTERS
3382      THIS TEST WILL USE THE LOWEST USEABLE MAP REGISTER.
3383      IF AN ERROR OCCURS THE TEST WILL REPORT THE PHYSICAL ADDRESS
3384      THAT WAS DESIRED, AND THE DATA AT THE ADDRESS THAT WAS REFERENCED.
3385 *****
3386
3387 TST30:
3388 015676 000004                    SCOPE
3389 015700 012737 016612 001324      MOV      #TST31,NXTTST ;SAVE STARTING ADDRESS OF NEXT TEST
3390                                     ;FOR ESCAPE ON PARITY ERRORS
3391 015706 052737 000040 172516      BIS      #BITS,MMR3    ;TURN ON MAP RELOCATION
3392
3393      ;THE RELOCATION HERE USES A BASE OF 00060000 AND AN
3394      ;OFFSET OF 00000 TO PRODUCE AN ADDRESS OF 00060000
3395
3396 015714 012737 015746 001110 100$: MOV      #1$,SLPERR   ;SET LOOP ON ERROR POINTER TO 1$
3397 015722 005077 163320                    CLR      @LREGU        ;CLEAR UPPER BITS OF MAPPING REG
3398 015726 012777 060000 163310      MOV      #060000,@LREGL ;LOAD LOWER BITS OF MAPPING REG
3399 015734 013737 001240 172354      MOV      LOWEST,@MKIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
3400 015742 012700 140000                    MOV      #140000,R0    ;SELECT PAR6, OFFSET IS 00000
3401 015746 000240                    1$:  NOP
3402 015750 011001                    MOV      (R0),R1       ;THIS IS A SYNC POINT FOR SCOPING
3403 015752 012702 060000                    MOV      #060000,R2    ;READ LOCATION 060000 THRU THE UNIBUS
3404 015756 020102                    CMP      R1,R2         ;THE EXPECTED PHYSICAL ADDRESS IS 060000
3405 015760 001401                    BEQ      2$           ;SEE IF THE MAP'S FETCH WAS CORRECT
3406 015762 104041                    ERROR   41             ;BRANCH IF FETCHED DATA MATCHES ADDRESS
3407                                     ;FAULTY RELOCATION BY UNIBUS MAP
3408
3409      ;THE RELOCATION HERE USES A BASE OF 00052524 AND AN
3410      ;OFFSET OF 05252 TO PRODUCE AN ADDRESS OF 00057776
3411
3412 015764 012737 016016 001110 2$:  MOV      #3$,SLPERR   ;SET LOOP ON ERROR POINTER TO 3$
3413 015772 005077 163250                    CLR      @LREGU        ;CLEAR UPPER BITS OF MAPPING REG
3414 015776 012777 052524 163240      MOV      #052524,@LREGL ;LOAD LOWER BITS OF MAPPING REG
3415 016004 013737 001240 172354      MOV      LOWEST,@MKIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
3416 016012 012700 145252                    MOV      #145252,R0    ;SELECT PAR6, OFFSET IS 05252
3417 016016 000240                    3$:  NOP
3418 016020 011001                    MOV      (R0),R1       ;THIS IS A SYNC POINT FOR SCOPING
3419 016022 012702 057776                    MOV      #057776,R2    ;READ LOCATION 057776 THRU THE UNIBUS
3420 016026 020102                    CMP      R1,R2         ;THE EXPECTED PHYSICAL ADDRESS IS 057776
3420 016030 001401                    BEQ      4$           ;SEE IF THE MAP'S FETCH WAS CORRECT
3420                                     ;BRANCH IF FETCHED DATA MATCHES ADDRESS
```



```

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$:  MOV      #13$,SLPERR      ;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          NOP
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$:  MOV      #15$,SLPERR      ;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          NOP
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
3510 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$:  MOV      #17$,SLPERR      ;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          NOP
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$:  MOV      #19$,SLPERR      ;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 0. 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
```

::
*THE RELOCATION HERE USES A BASE OF 00057776 AND AN
*OFFSET OF 00002 TO PRODUCE AN ADDRESS OF 00060000

```
3542  
3543  
3544  
3545  
3546 016534 012737 016566 001110 20$: MOV #21$,SLPERR ;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$: MOV #100$,SLPERR ;SET LOOP POINTER TO START OF TEST
```

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

```
3560  
3561  
3562  
3563  
3564  
3565  
3566  
3567  
3568  
3569  
3570 016612  
3571 016612 000004  
3572 016614 012737 017116 001324 TST31: SCOPE  
3573  
3574 016622 012737 000012 001204 MOV #TST32,NXTTST ;SAVE STARTING ADDRESS OF NEXT TEST  
3575  
3576  
3577 016630 012737 177777 016764 20$: MOV #-1,4$ ;FOR ESCAPE ON PARITY ERRORS  
3578 016636 005077 162404 CLR @LREGU ;DO 12 ITERATIONS  
3579 016642 012777 020000 162374 MOV #20000,@LREGL ;BIT05 IS NON-EXISTANT MEMORY BIT IN  
3580 016650 012701 100100 MOV #100100,R1 ;THE CPU ERROR REGISTER  
3581 016654 012700 150000 MOV #150000,R0 ;INITIALIZE FLAG AS NEGATIVE ONE  
3582 016660 012737 000277 172350 MOV #277,KIPAR4 ;CLEAR UPPER 6 BITS OF MAP REG  
3583 016666 013737 001240 172354 MOV LOWEST,KIPAR6 ;LOAD 4K BASE INTO MAP REGISTER  
3584 016674 012737 016734 001110 MOV #10$,SLPERR ;LOAD BITS TO SELECT PAR 4, OFFSET 100  
3585 016702 012737 000020 001264 3$: MOV #20,(PUEXP ;LOAD BITS TO SELECT PAR 6, OFFSET 2K  
3586 016710 005037 001266 CLR PCPJR ;START WITH PHYSICAL 6K  
3587 016714 013710 001326 MOV DATA,(R0) ;LOAD PAR 6 WITH MAP REG'S ADDR  
3588 ;SET LOOP ON ERROR POINTER TO 10$  
;EXPECTING A UNIBUS TIME OUT DURING TEST  
;CLEAR TIME OUT FLAG  
;THIS LOAD WILL TIME OUT WHEN YOU  
;HAVE REACHED THE TOP OF MEMORY
```

```

3589
3590
3591
3592
3593
3594 016720 005737 001266
3595 016724 001016
3596 016726 012737 000040 001264
3597 016734 011103
3598 016736 022737 000040 001266
3599 016744 001414
3600 016746 000240
3601 016750 011002
3602 016752 020203
3603
3604 016754 001410
3605 016756 104042
3606 016760 000406
3607 016762 005227
3608 016764 177777
3609 016766 001003
3610 016770 013737 172350 001320
3611 016776 062737 000100 001326
3612 017004 062737 000100 172350
3613 017012 062777 010000 162224
3614 017020 001330
3615 017022 005277 162220
3616 017026 022777 000073 162212
3617 017034 103322
3618 017036 005237 001326
3619 017042 042737 177700 001326
3620 017050 052737 000300 001326
3621 017056 012737 016630 001110
3622 017064 023737 177760 001320
3623 017072 001407
3624
3625 017074 105737 001331
3626 017100 001004
3627 017102 105737 001332
3628 017106 001001
3629 017110 104043
3630 017112 005037 001264

```

10S:

1S:

4S:

2S:

19S:

```

TST PCPUER
BNE 1$
MOV #NEXMEM,CPUEXP
MOV (R1),R3
CMP #NEXMEM,PCPUER
BEQ 2$
NOP
MOV (R0),R2
CMP R2,R3

BEQ 2$
ERROR 42
BR 2$
INC (PC)+
WORD -1
BNE 2$
MOV KIPAR4,RSIZE
ADD #100,DATA
ADD #100,KIPAR4
ADD #10000,@ALREG
BNE 3$
INC @ALREGU
CMP #73,@ALREGU
BHS 3$
INC DATA
BIC #177700,DATA
BIS #300,DATA
MOV #20$,SLPERR
CMP @SIZELO,RSIZE
BEQ 19$

TSTB KB11EM
BNE 19$
TSTB KB11CM
BNE 19$
ERROR 43
CLR CPUEXP

```

```

: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.
:SEE IF THERE WAS MAIN MEMORY
:BRANCH IF NO MAIN MEMORY FROM UNIBUS
:POSSIBLE CACHE NON-EXISTANT MEMORY
:READ TEST LOCATION VIA FASTBUS
:WAS THIS CACHE NON-EXISTANT MEMORY
:BRANCH IF NON-EXISTANT MEMORY
:THIS IS A SYNC POINT FOR SCOPING
:READ TEST LOCATION VIA UNIBUS MAP
:COMPARE TEST DATA R2=MAP DATA
:R3=FASTBUS DATA
:BRANCH IF IT WAS THE SAME
:CARRY PROPAGATION FAILURE IN MAP
:BRANCH TO UPDATE ROUTINE
:INCREMENT ONE TIME ENTRANCE FLAG
:USE NEGATIVE ONE FOR FLAG
:BRANCH IF YOU'VE BEEN HERE BEFORE
:SAVE UPPER LIMIT OF MEMORY
:CHANGE PATTERN FOR NEXT LOAD
:ADD 2K TO PAR4
:ADD 2K TO MAP REGISTER
:BRANCH IF MAP REGISTER NOT ZERO
:ADD ONE TO UPPER 6 BITS OF MAP REG
:SEE IF TOP 128K BLOCK HAS BEEN PASSED
:BRANCH IF NOT PAST IT
:CHANGE DATA PATTERN FOR NEXT PASS
:CLEAR UPPER 10 BITS OF DATA PATTERN
:START WITH 3XX IN DATA PATTERN
:SET LOOP POINTER TO START OF TEST
:SEE IF SIZE JUMPERS AGREE WITH MEMORY TOP
:BRANCH IF TOP OF MEMORY AGREES WITH
:THE SIZE JUMPERS
:RUNNING ON A KB11-EM?
:SKIP ERROR IF YES
:RUNNING ON MODIFIED MACHINE?
:BRANCH IF YES
:TOP OF MEMORY DOESN'T MATCH SIZE JUMPER
:NO CPU TRAPS EXPECTED FOR AWHILE

```

```

3631
3632
3633
3634
3635
3636
3637
3638
3639
3640
3641
3642
3643
3644

```

```

*****
:TEST 32 PARITY REPORTING THRU THE MAP, MAIN MEMORY EVEN WORD
:
:THIS TEST FORCES A PARITY ERROR IN MAIN MEMORY AT PHYSICAL
:ADDRESS 00040000 BY SETTING ONE BIT IN EACH BYTE OF THE
:WORD. THE TEST THEN FORCES THE EVEN WORD PARITY BITS TO
:ONES AND READS ADDRESS 00040000 THRU THE UNIBUS MAP. THEN
:IT CHECKS TO SEE THAT THE PARITY ABORT OCCURRED AND THAT THE
:CONDITION WAS CORRECT BEFORE GOING TO THE NEXT TEST.
:
:ERRORS ARE REPORTED INLINE IN THE TEST CODE. AND THE PARITY

```

3645
3646
3647
3648 017116
3649 017116 000004
3650 017120 012737 017360 001324
3651
3652 017126 104416
3653 017130 012737 023404 001200 20\$:
3654 017136 005037 001274
3655 017142 012737 177777 177744
3656 017150 013737 177746 001202
3657 017156 012737 000014 177746
3658 017164 013737 001240 172354
3659 017172 012777 040000 162044
3660 017200 005077 162042
3661 017204 012701 177750
3662 017210 012737 017234 001110
3663 017216 012700 140000
3664 017222 012710 020001
3665 017226 012737 017320 000114
3666 017234 012737 030000 177750 1\$:
3667 017242 000240
3668 017244 011003
3669 017246 011102
3670 017250 005011
3671 017252 050000
3672 017254 023737 001200 001274
3673 017262 001401
3674 017264 104044
3675 017266 012737 177777 177744 2\$:
3676 017274 012737 005764 000114
3677 017302 013737 001202 177746
3678 017310 012737 017130 001110
3679 017316 000420
3680
3681
3682 017320 013737 177740 001270 10\$:
3683 017326 013737 177742 001272
3684 017334 013737 177744 001274
3685 017342 013737 177746 001276
3686 017350 012737 030000 001300
3687 017356 000002
3688
3689
3690
3691
3692
3693
3694
3695
3696
3697
3698
3699
3700

```

: * VECTOR WILL TRAP TO LABEL 10$ AT THE END OF THIS TEST.
: *
: * *****
: * TST32:
: *
: * SCOPE
: * MOV #1ST33,NXTTST ;SAVE STARTING ADDRESS OF NEXT TEST
: * ;FOR ESCAPE ON PARITY ERRORS
: * ;TURN OFF T BIT TRAPPING IF ON
: * ;EXPECTED ERROR CONDITION
: * ;CLEAR MEM ERROR REG STORAGE
: * ;CLEAR MEMORY ERROR REGISTER
: * ;SAVE CONTROL REG TO RESTORE CACHE
: * ;FORCE MISSES IN CACHE
: * ;PUT UNIBUS ADDR OF MAP REG IN PAR6
: * ;LOWEST MAP REGISTER POINTS TO 8K
: * ;CLEAR UPPER BITS OF MAP REG
: * ;PUT ADDR OF MAINT REG IN R1
: * ;SET LOOP ON ERROR POINTER TO 1$
: * ;SELECT KIPAR6, OFFSET 0
: * ;LOAD ONE BIT IN EACH BYTE LF 40000
: * ;SET PARITY VECTOR TO 10$
: * ;FORCE EVEN WORD PARITY BITS TO ONE
: * ;THIS IS A SYNC POINT FOR SCOPING
: * ;'DATA FETCH' SHOULD CAUSE PARITY ABORT
: * ;SAVE MAINT REG IN CASE NO TRAP
: * ;CLEAR MAINT REG IN CASE NO TRAP
: * ;DUPPY INST WITH P.B.'S ON
: * ;SEE IF ERROR REG HOLDS RIGHT DATA
: * ;BRANCH IF CONDITION WAS CORRECT
: * ;PARITY REPORTING BAD
: * ;CLEAR MEMORY ERROR REGISTER
: * ;RESTORE PARITY SERVICE ROUTINE
: * ;RESTORE CACHE TO FORMER CONDITION
: * ;SET LOOP POINTER TO START OF TEST
: * ;TEST OVER GO TO NEXT TEST
: *
: *
: * ;SAVE LOWER CACHE ADDR REG
: * ;SAVE HIGH BITS OF FAILING ADDR
: * ;SAVE MEMORY ERROR REGISTER
: * ;SAVE CONTROL REGISTER FOR TYPE OUT
: * ;SAVE DATA IN MAINTENANCE REGISTER
: * ;RETURN TO TEST AND CHECK PARITY
: *
: * *****
: * *TEST 33 PARITY REPORTING THRU THE MAP, MAIN MEMORY ODD WORD
: *
: * THIS TEST FORCES A PARITY ERROR IN MAIN MEMORY AT PHYSICAL
: * ADDRESS 00040002 BY SETTING ONE BIT IN EACH BYTE OF THE
: * WORD. THE TEST THEN FORCES THE ODD WORD PARITY BITS TO
: * ONES AND READS ADDRESS 00040002 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

```

3701
3702
3703
3704 017360
3705 017360 000004
3706 017362 012737 017632 001324
3707
3708 017370 012737 023410 001200 20\$:
3709 017376 005037 001274
3710 017402 012737 177777 177744
3711 017410 013737 177746 001202
3712 017416 012737 000014 177746
3713 017424 013737 001240 172354
3714 017432 012777 040000 161604
3715 017440 005077 161602
3716 017444 012701 177750
3717 017450 012737 017506 001110
3718 017456 005037 140000
3719 017462 012700 140002
3720 017466 012737 017506 001110
3721 017474 012710 020001
3722 017500 012737 017572 000114
3723 017506 012737 140000 177750 1\$:
3724 017514 000240
3725 017516 011003
3726 017520 011102
3727 017522 005011
3728 017524 050000
3729 017526 023737 001200 001274
3730 017534 001401
3731 017536 104044
3732 017540 012737 177777 177744 2\$:
3733 017546 012737 005764 000114
3734 017554 013737 001202 177746
3735 017562 012737 017370 001110
3736 017570 000420
3737
3738
3739 017572 013737 177740 001270 10\$:
3740 017600 013737 177742 001272
3741 017606 013737 177744 001274
3742 017614 013737 177746 001276
3743 017622 012737 140000 001300
3744 017630 000002
3745
3746
3747
3748
3749
3750
3751
3752
3753
3754
3755
3756

VECTOR WILL TRAP TO LABEL 10\$ AT THE END OF THIS TEST.

TST33:
SCOPE
MOV #TST34,NXTTST ;SAVE STARTING ADDRESS OF NEXT TEST
;FOR ESCAPE ON PARITY ERRORS
MOV #23410,\$TMP4 ;LOAD EXPECTED ERROR CONDITION
CLR PPARER ;CLEAR MEM ERROR REG STORAGE
MOV #-1,@MEMERR ;CLEAR MEMORY ERROR REGISTER
MOV @CONTRL,\$TMP5 ;SAVE CONTROL REG TO RESTORE CACHE
MOV #14,@CONTRL ;FORCE MISSES IN CACHE
MOV LOWEST,KIPAR6 ;PUT UNIBUS ADDR OF MAP REG IN PAR6
MOV #40000,@LREGL ;LOWEST MAP REGISTER POINTS TO 8K
CLR @LREGU ;CLEAR UPPER BITS OF MAP REG
MOV @MAINT,R1 ;PUT ADDR OF MAINT REG IN R1
MOV #1\$,\$LPERR ;SET LOOP ON ERROR POINTER TO 1\$
CLR 140000 ;CLEAR LOCATION 40000
MOV #140002,R0 ;LOAD ADDRESS OF 40002 INTO R0
MOV #1\$,\$LPERR ;SET LOOP ON ERROR POINTER TO 1\$
MOV #20001,(R0) ;LOAD ONE BIT IN EACH BYTE OF 40002
MOV #10\$,\$CACHVEC ;SET PARITY VECTOR TO 10\$
MOV #140000,@MAINT ;FORCE ODD WORD PARITY BITS TO ONE
NOP ;THIS IS A SYNC POINT FOR SCOPING
MOV (R0),R3 ;'DATA FETCH' SHOULD CAUSE PARITY ABORT
MOV (R1),R2 ;SAVE MAINT REG IN CASE NO TRAP
CLR (R1) ;CLEAR MAINT REG IN CASE NO TRAP
BIS R0,R0 ;DUPPLY INST WITH P.B.'S ON
CMP \$TMP4,PPARER ;SEE IF MEM ERROR REG HOLDS RIGHT DATA
BEQ 2\$;BRANCH IF ERROR WAS EXPECTED ONE
ERROR 44 ;PARITY REPORTING BAD
MOV #-1,MEMERR ;CLEAR MEMORY ERROR REGISTER
MOV @MEMERR,\$CACHVEC ;RESTORE PARITY SERVICE ROUTINE
MOV \$TMP5,CONTRL ;RESTORE CACHE TO FORMER CONDITION
MOV #20\$,\$LPERR ;SET LOOP POINTER TO START OF TEST
BR TST34 ;TEST IS OVER, GO TO NEXT TEST

MOV @LOADRS,\$LOADR ;SAVE LOWER CACHE ADDR REG
MOV @HIADRS,\$HIADR ;SAVE HIGH BITS OF FAILING ADDR
MOV @MEMERR,PPARER ;SAVE MEMORY ERROR REGISTER
MOV @CONTRL,\$CONTR ;SAVE CONTROL REGISTER FOR TYPE OUT
MOV #140000,\$MAINT ;SAVE DATA IN MAINTENANCE REGISTER
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.

3757
3758
3759
3760
3761
3762
3763
3764
3765
3766
3767
3768
3769
3770
3771
3772
3773
3774
3775
3776
3777
3778
3779
3780
3781
3782
3783
3784
3785
3786
3787
3788
3789
3790
3791
3792
3793
3794
3795
3796
3797
3798
3799
3800
3801
3802
3803
3804
3805
3806
3807
3808
3809
3810
3811
3812

017632
017632 000004
017634 012737 020120 001324
017642 012737 177777 020006
017650 032737 000004 177746
017656 001120
017660 013737 177746 001202
017666 012737 023504 001200
017674 005037 001274
017700 012737 177777 177744
017706 013737 001240 172354
017714 005077 161326
017720 012777 040010 161316
017726 012705 177750
017732 012704 140000
017736 012714 100000

017742 012737 000030 177746
017750 011401
017752 005001
017754 012737 020060 000114
017762 010137 177750
017766 000241

017770 011401
017772 011507
017774 1C
017776 006.
020000 012701 001020

020004 005227
020006 177777
020010 001764
020012 023737 001200 001274
020020 001402
020022 010302
020024 104044
020026 012737 177777 177744
020034 012737 005764 000114
020042 013737 001202 177746
020050 012737 017642 001110
020056 000420

020060 013737 177740 001270
020066 013737 177742 001272
020074 013737 177744 001274

```
*****  
: * ERRORS ARE REPORTED INLINE IN THE TEST CODE, AND THE PARITY  
: * VECTOR WILL TRAP TO LABEL 10$ AT THE END OF THIS TEST.  
: *  
: *****  
: TST34:  
: SCOPE  
: MOV #TST35,NXTTST ;SAVE STARTING ADDRESS OF NEXT TEST  
: ;FOR ESCAPE ON PARITY ERRORS  
20$: MOV #-1,21$ ;RESTORE NEG ONE FOR NEXT ITERATION  
: BIT #BIT2,@#CONTRL ;SEE IF GROUP 0 IS DISABLED  
: BNE TST35 ;BRANCH IF GROUP 0 IS DISABLED  
: MOV CONTRL,$TMP5 ;SAVE CONDITION OF CACHE  
: MOV #23504,$TMP4 ;EXPECTED PARITY CONDITION GROUP 0  
: CLR PPARER ;CLEAR MEM ERROR REG STORAGE  
: MOV #-1,@MEMERR ;CLEAR MEMORY ERROR REGISTER  
: MOV LOWEST,KIPAR6 ;PUT UNIBUS ADDR OF MAP REG IN PARS  
: CLR @LREGU ;CLEAR UPPER BITS ON MAP REG  
: MOV #40010,@LREGU ;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 #30,@#CONTRL ;FORCE SELECTION OF GROUP 0 ON READ  
: MOV (R4),R1 ;PUT (40000) & (40002) IN GROUP 0  
: CLR R1 ;ZERO R1 FOR THE MOMENT  
: MOV #10$,CACHVEC ;SET PARITY VECTOR TO 10$  
1$: MOV R1,@MAINT ;FORCE GPO PARITY BITS TO 0 ON 2ND PASS  
: CLR ;THIS IS A SYNC POINT FOR SCOPING  
: ;THE LOW BYTE HAS ITS P.B. OFF  
: MOV (R4),R1 ;DATA FETCH SHOULD CAUSE PARITY ABORT  
: MOV (R5),R3 ;SAVE MAINT REG IN CASE NO TRAP  
: CLR R5 ;CLEAR MAINT REG IN CASE NO TRAP  
: SXT R1 ;DUPPY INST WITH P.B.'S OFF  
: MOV #020020,R1 ;FORCE GROUP 0 LOW BYTE PARITY BIT  
: ;TO 0 AND MAIN MEMORY EVEN WORD HIGH  
: ;BYTE PARITY BIT TO 1  
: ;INCREMENT NEXT WORD TO ZERO ON FIRST PASS  
21$: INC (PC)+  
: .WORD -1  
: BEQ 1$ ;BRANCH ON FIRST PASS ONLY  
: CMP $TMP4,PPARER ;SEE IF ERROR CONDITION MATCHES EXPECTED  
: BEQ 2$ ;BRANCH IF CORRECT CONDITION  
: MOV R3,R2 ;R2 HOLDS MAINT REG FOR ERROR TYPE OUT  
: ERROR 44 ;PARITY REPORTING BAD  
2$: MOV #-1,MEMERR ;CLEAR MEMORY ERROR REGISTER  
: MOV @MEMER,CACHVEC ;RESTORE PARITY SERVICE ROUTINE  
: MOV $TMP5,CONTRL ;RESTORE CACHE TO FORMER CONDITION  
: MOV #20$,LPERR ;SET LOOP POINTER TO START OF TEST  
: BR TST35 ;TEST OVER GO TO NEXT TEST  
  
10$: MOV @PLOADRS,PLOADR ;SAVE LOWER CACHE ADDR REG  
: MOV @PHIADRS,PHIADR ;SAVE HIGH BITS OF FAILING ADDR  
: MOV @MEMERR,PPARER ;SAVE MEMORY ERROR REGISTER
```

3813 020102 013737 177746 001276
3814 020110 012737 030060 001300
3815 020116 000002
3816
3817
3818
3819
3820
3821
3822
3823
3824
3825
3826
3827
3828
3829
3830
3831
3832

MOV @#CONTRL,PCONTR ;SAVE CONTROL REGISTER FOR TYPE OUT
MOV #30060,PMANT ;SAVE DATA IN MAINTENANCE 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:

3833 020120
3834 020120 000004
3835 020122 012737 020464 001324
3836
3837 020130 012737 177777 020274 20\$:
3838 020136 032737 000010 177746
3839 020144 001120
3840 020146 013737 177746 001202
3841 020154 012737 023604 001200
3842 020162 005037 001274
3843 020166 012737 177777 177744
3844 020174 013737 001240 172354
3845 020202 005077 161040
3846 020206 012777 040000 161030
3847 020214 012705 177750
3848 020220 012704 140000
3849 020224 012714 100000
3850
3851
3852
3853 020230 012737 000044 177746
3854 020236 011401
3855 020240 005001
3856 020242 012737 020346 000114 1\$:
3857 020250 010137 177750
3858 020254 000241
3859
3860 020256 011401
3861 020260 011503
3862 020262 105015
3863 020264 006701
3864 020266 012701 020100
3865
3866
3867 020272 005227
3868 020274 177777 21\$:

SCOPE
MOV #TST36,NXTTST ;SAVE STARTING ADDRESS OF NEXT TEST
;FOR ESCAPE ON PARITY ERRORS
MOV #-1,21\$;RESTORE NEG ONE FOR NEXT ITERATION
BIT #BIT3,CONTRL ;SEE IF GROUP 1 IS DISABLED
BNE U\$MAP ;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,@LREGU ;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\$
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 ;DUPLY 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
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,MEMERH ;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 UEMAP ;;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 MAINTENANCE REGISTER
3886 020404 000002 RTI ;RETURN TO TEST AND CHECK PARITY
3887
3888
3889
```

```
3890 .SBTTL *****
3891 .SBTTL THE FOLLOWING TESTS ARE RUN THROUGH THE UNIBUS MAP
3892 .SBTTL *****
3893
```

```
3894
3895 020406 000004 UEMAP: SCOPE ;LOOP ON PREVIOUS TEST
3896 020410 104420 TBITR ;RESTORE T-BIT IF IT WAS ON
3897
```

```
3898 :
3899 : THIS CODE SETS UP THE TWO MAP REGISTERS ABOVE THE LOWEST
3900 : USEABLE ONE TO POINT TO PHYSICAL MEMORY FROM 0 - 8K. IN THE
3901 : DEFAULT CASE, IN MANUFACTURING AND IF THERE IS NO UNIBUS MEMORY,
3902 : THIS WILL BE MAP REGISTERS 1 AND 2. MAP REGISTER 1 WILL POINT
3903 : TO PHYSICAL 4K - 8K SO 'ACT-11' WILL WORK PROPERLY AND MAP
3904 : REGISTER 2 WILL POINT TO PHYSICAL 0 - 4K. THIS MEANS THAT
3905 : KIPARO SHOULD GET 170400 SO IT PUTS ADDRESSES 040000 TO 057776
3906 : ON THE UNIBUS AND KIPAR1 SHOULD GET 170200 SO IT PUTS ADDRESSES
3907 : 020000 TO 037776 ON THE UNIBUS.
```

```
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 KIPARO
3918 020460 010137 172340 MOV R1,KIPARO ;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
```

```

3925
3926
3927
3928
3929
3930
3931
3932
3933
3934
3935 020464
3936 020464 000004
3937 020466 012737 020540 001106
3938 020474 012737 020540 001110
3939 020502 012737 000036 001102
3940 020510 013737 001102 177570
3941 020516 012737 020634 001324
3942 020524 022737 167777 177760
3943 020532 103002
3944 020534 000137 020634
3945 020540 012737 000020 001264 20$:
3946 020546 013737 001240 172350
3947 020554 012777 000074 160464
3948 020562 012777 000000 160454
3949 020570 012737 020576 001110
3950 020576 005037 001266 1$:
3951 020602 000240
3952 020604 013703 100000
3953
3954
3955
3956
3957
3958
3959 020610 022737 000020 001266
3960 020616 001401
3961 020620 104045
3962 020622 012737 020540 001110 10$:
3963 020630 005037 001264
3964
3965
3966
3967
3968
3969
3970
3971
3972
3973
3974
3975
3976
3977
3978
3979
3980 020634

```

```

: * 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.
: *
: * *****

```

```

TST36:
SCOPE
MOV #20$, $LPADR ;SET LOOP ON TEST POINTER TO 20$
MOV #20$, $LPERR ;SET LOOP ON ERROR POINTER TO 20$
MOV #36, $TSTNM ;SETUP TEST NUMBER AND CLR ERROR FLAG
MOV $TSTNM, DISPLAY ;DISPLAY TEST NUMBER FOR ALL TO SEE
MOV #TST37, NXTTST ;SET UP ESCAPE VECTOR IN CASE OF PARITY ERRORS
CMP #167777, SIZELO ;>1920K?
BHS 20$ ;BRANCH IF NOT
JMP TST37 ;ELSE SKIP TEST
MOV #20, CPUEXP ;EXPECTING CPU TIMEOUT IN THIS TEST
MOV LOWEST, KIPAR4 ;LOAD PAR 4 WITH LOWEST USABLE MAP REG
MOV #74, @LREGU ;LOAD UPPER 6 BITS OF LOWEST MAP REG
MOV #000000, @LREGL ;LOAD LOWER 16 BITS OF LOWEST MAP REG
MOV #1$, $LPERR ;SET LOOP ON ERROR POINTER TO 1$
CLR PCPUER ;CPU ERROR REG LOCATION
NOP ;THIS IS A SYNC POINT FOR SCOPING
MOV @#170000, 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.
;THE UNIBUS SHOULD HAVE TIMED OUT
CMP #20, PCPUER ;BRANCH IF UNIBUS TIMED OUT
BEQ 10$ ;DID NOT TIME OUT OVER UNIBUS
ERROR 45 ;SET LOOP POINTER TO START OF TEST
MOV #20$, $LPERR ;NO CPU TRAPS EXPECTED IN NEXT TEST
CLR CPUEXP

```

```

: * *****
: * 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.
: *
: * *****

```

TST37:

```
3981 020634 000004          SCOPE
3982 020636 012737 021550 001324 MOV #TST40,NXTTST ;SAVE STARTING ADDRESS OF NEXT TEST
3983                                     ;FOR ESCAPE ON PARITY ERRORS
3984 020644 012737 060000 060000 MOV #060000,@#060000 ;MAKE SURE THAT ADDRESS 060000
3985                                     ;CONTAINS ITS OWN ADDRESS AS DATA
3986
3987 ::
3988 ::*THE RELOCATION HERE USES A BASE OF 00060000 AND AN
3989 ::*OFFSET OF 0000 TO PRODUCE AN ADDRESS OF 00060000
3990 020652 012737 020704 001110 100$: MOV #1$,SLPERR ;SET LOOP ON ERROR POINTER TO 1$
3991 020660 005077 160362          CLR @LR$GU ;CLEAR UPPER BITS OF MAPPING REG
3992 020664 012777 060000 160352 MOV #060000,@LR$GL ;LOAD LOWER BITS OF MAPPING REG
3993 020672 013737 001240 172354 MOV LOWEST,@#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
3994 020700 012700 140000          MOV #140000,R0 ;SELECT PAR6, OFFSET IS 00000
3995 020704 000240          1$: NOP ;THIS IS A SYNC POINT FOR SCOPING
3996 020706 011001          MOV (R0),R1 ;READ LOCATION 060000 THRU THE UNIBUS
3997 020710 012702 060000          MOV #060000,R2 ;THE EXPECTED PHYSICAL ADDRESS IS 060000
3998 020714 020102          CMP R1,R2 ;SEE IF THE MAP'S FETCH WAS CORRECT
3999 020716 001401          BEQ 2$ ;BRANCH IF FETCHED DATA MATCHES ADDRESS
4000 020720 104046          ERROR 46 ;FAULTY RELOCATION BY UNIBUS MAP
4001
4002 ::
4003 ::*THE RELOCATION HERE USES A BASE OF 00052524 AND AN
4004 ::*OFFSET OF 05252 TO PRODUCE AN ADDRESS OF 00057776
4005 020722 012737 020754 001110 2$: MOV #3$,SLPERR ;SET LOOP ON ERROR POINTER TO 3$
4006 020730 005077 160312          CLR @LR$GU ;CLEAR UPPER BITS OF MAPPING REG
4007 020734 012777 052524 160302 MOV #052524,@LR$GL ;LOAD LOWER BITS OF MAPPING REG
4008 020742 013737 001240 172354 MOV LOWEST,@#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
4009 020750 012700 145252          MOV #145252,R0 ;SELECT PAR6, OFFSET IS 05252
4010 020754 000240          3$: NOP ;THIS IS A SYNC POINT FOR SCOPING
4011 020756 011001          MOV (R0),R1 ;READ LOCATION 057776 THRU THE UNIBUS
4012 020760 012702 057776          MOV #057776,R2 ;THE EXPECTED PHYSICAL ADDRESS IS 057776
4013 020764 020102          CMP R1,R2 ;SEE IF THE MAP'S FETCH WAS CORRECT
4014 020766 001401          BEQ 4$ ;BRANCH IF FETCHED DATA MATCHES ADDRESS
4015 020770 104046          ERROR 46 ;FAULTY RELOCATION BY UNIBUS MAP
4016
4017 ::
4018 ::*THE RELOCATION HERE USES A BASE OF 00045252 AND AN
4019 ::*OFFSET OF 12524 TO PRODUCE AN ADDRESS OF 00057776
4020 020772 012737 021024 001110 4$: MOV #5$,SLPERR ;SET LOOP ON ERROR POINTER TO 5$
4021 021000 005077 160242          CLR @LR$GU ;CLEAR UPPER BITS OF MAPPING REG
4022 021004 012777 045252 160232 MOV #045252,@LR$GL ;LOAD LOWER BITS OF MAPPING REG
4023 021012 013737 001240 172354 MOV LOWEST,@#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
4024 021020 012700 152524          MOV #152524,R0 ;SELECT PAR6, OFFSET IS 12524
4025 021024 000240          5$: NOP ;THIS IS A SYNC POINT FOR SCOPING
4026 021026 011001          MOV (R0),R1 ;READ LOCATION 057776 THRU THE UNIBUS
4027 021030 012702 057776          MOV #057776,R2 ;THE EXPECTED PHYSICAL ADDRESS IS 057776
4028 021034 020102          CMP R1,R2 ;SEE IF THE MAP'S FETCH WAS CORRECT
4029 021036 001401          BEQ 6$ ;BRANCH IF FETCHED DATA MATCHES ADDRESS
4030 021040 104046          ERROR 46 ;FAULTY RELOCATION BY UNIBUS MAP
4031
4032 ::
4033 ::*THE RELOCATION HERE USES A BASE OF 00050420 AND AN
4034 ::*OFFSET OF 10420 TO PRODUCE AN ADDRESS OF 00061040
4035 021042 012737 021074 001110 6$: MOV #7$,SLPERR ;SET LOOP ON ERROR POINTER TO 7$
4036 021050 005077 160172          CLR @LR$GU ;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
    
```

```

4149 021536 001401      BEQ      22$      ;BRANCH IF FETCHED DATA MATCHES ADDRESS
4150 021540 104046      ERROR    46      ;FAULTY RELOCATION BY UNIBUS MAP
4151 021542 012737 020652 001110 22$:  MOV      #100$, $LPERR ;SET LOOP POINTER TO START OF TEST
4152
4153
4154
4155
4156
4157
4158
4159
4160
4161
4162
4163
4164
4165
4166 021550
4167 021550 000004
4168 021552 012737 022054 001324
4169
4170 021560 012737 000012 001204
4171 021566 012737 177777 021722 20$:  MOV      #12,$TIMES ;DO 12 ITERATIONS
4172 021574 005077 157446      MOV      #-1,$S ;INITIALIZE FLAG AS NEGATIVE ONE
4173 021600 012777 020000 157436      CLR      @LREGU ;CLEAR UPPER 6 BITS OF MAP REG
4174 021606 012701 100100      MOV      #20000,@LREGL ;LOAD 4K BASE INTO MAP REGISTER
4175 021612 012700 150000      MOV      #100100,R1 ;LOAD BITS TO SELECT PAR 4, OFFSET 100
4176 021616 012737 000277 172350      MOV      #150000,R0 ;LOAD BITS TO SELECT PAR 6, OFFSET 2K
4177 021624 013737 001240 172354      MOV      #277,KIPAR4 ;START WITH PHYSICAL 6K
4178 021632 012737 021672 001110      MOV      LOWEST,KIPAR6 ;LOAD PAR 6 WITH MAP REG'S ADDR
4179 021640 012737 000020 001264 3$:  MOV      #10$, $LPERR ;SET LOOP ON ERROR POINTER TO 10$
4180 021646 005037 001266      MOV      #TIMOUT,$CPUEXP ;EXPECTING A UNIBUS TIME OUT DURING TEST
4181 021652 013710 001326      CLR      PCPUER ;CLEAR TIME OUT FLAG
4182
4183
4184
4185
4186
4187
4188 021656 005737 001266      MOV      DATA,(R0) ;THIS LOAD WILL TIME OUT WHEN YOU
4189 021662 001016      TST      PCPUER ;HAVE REACHED THE TOP OF MEMORY
4190 021664 012737 000040 001264 10$:  BNE      1$ ;IT SELECTS PAR 6 WHICH WILL PUT ADDR
4191 021672 011103      MOV      #NEXMEM,$CPUEXP ;<XXX XXI>0000 ON THE UNIBUS.
4192 021674 022737 000040 001266      MOV      (R1),R3 ;THE X'S WILL SELECT THE LOWEST USEABLE
4193 021702 001414      CMP      #NEXMEM,$PCPUER ;MAPPING REGISTER. THE DEFAULT CASE IS
4194 021704 000240      BEQ      2$ ;010000, SELECTING MAPPING REGISTER 0.
4195 021706 011002      NOP ;SEE IF THERE WAS MAIN MEMORY
4196 021710 020203      MOV      (R0),R2 ;BRANCH IF NO MAIN MEMCRY FROM UNIBUS
4197
4198 021712 001410      MOV      R2,R3 ;POSSIBLE CACHE NON-EXISTANT MEMORY
4199 021714 104047      BEQ      2$ ;READ TEST LOCATION VIA FASTBUS
4200 021716 000406      BEQ      2$ ;WAS THERE CACHE NON-EXISTANT MEMORY
4201 021720 005227 1$:  INC      (PC)+ ;BRANCH IF NON-EXISTANT MEMORY
4202 021722 177777 4$:  .WORD   -1 ;THIS IS A SYNC POINT FOR SCOPING
4203 021724 001003      BNE      2$ ;READ TEST LOCATION VIA UNIBUS MAP
4204 021726 013737 172350 001320      MOV      KIPAR4,$RSIZE ;COMPARE TEST DATA R2=MAP DATA
                                        ;R3=FASTBUS DATA
                                        ;BRANCH IF IT WAS THE SAME
                                        ;FAILURE IN CARRY PROPAGATION IN MAP
                                        ;BRANCH TO UPDATE ROUTINE
                                        ;INCREMENT ONE TIME ENTRANCE FLAG
                                        ;USE NEGATIVE ONE FOR FLAG
                                        ;BRANCH IF YOU'VE BEEN HERE BEFORE
                                        ;SAVE UPPER LIMIT OF MEMORY

```

```

*****
*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.
*****

```

TST40:

```

SCOPE
MOV #TST41,NXTTST ;SAVE STARTING ADDRESS OF NEXT TEST
;FOR ESCAPE ON PARITY ERRORS
;DO 12 ITERATIONS
MOV #12,$TIMES
MOV #-1,$S ;INITIALIZE FLAG AS NEGATIVE ONE
CLR @LREGU ;CLEAR UPPER 6 BITS OF MAP REG
MOV #20000,@LREGL ;LOAD 4K BASE INTO MAP REGISTER
MOV #100100,R1 ;LOAD BITS TO SELECT PAR 4, OFFSET 100
MOV #150000,R0 ;LOAD BITS TO SELECT PAR 6, OFFSET 2K
MOV #277,KIPAR4 ;START WITH PHYSICAL 6K
MOV LOWEST,KIPAR6 ;LOAD PAR 6 WITH MAP REG'S ADDR
MOV #10$, $LPERR ;SET LOOP ON ERROR POINTER TO 10$
MOV #TIMOUT,$CPUEXP ;EXPECTING A UNIBUS TIME OUT DURING TEST
CLR PCPUER ;CLEAR TIME OUT FLAG
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 XXI>0000 ON THE UNIBUS.
;THE X'S WILL SELECT THE LOWEST USEABLE
;MAPPING REGISTER. THE DEFAULT CASE IS
;010000, SELECTING MAPPING REGISTER 0.
;SEE IF THERE WAS MAIN MEMORY
;BRANCH IF NO MAIN MEMCRY FROM UNIBUS
;POSSIBLE CACHE NON-EXISTANT MEMORY
;READ TEST LOCATION VIA FASTBUS
;WAS THERE CACHE NON-EXISTANT MEMORY
;BRANCH IF NON-EXISTANT MEMORY
;THIS IS A SYNC POINT FOR SCOPING
;READ TEST LOCATION VIA UNIBUS MAP
;COMPARE TEST DATA R2=MAP DATA
;R3=FASTBUS DATA
;BRANCH IF IT WAS THE SAME
;FAILURE IN CARRY PROPAGATION IN MAP
;BRANCH TO UPDATE ROUTINE
;INCREMENT ONE TIME ENTRANCE FLAG
;USE NEGATIVE ONE FOR FLAG
;BRANCH IF YOU'VE BEEN HERE BEFORE
;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\$,SLPERR	:SET LOOP POINTER TO START OF TEST
4216	022022	023737	177760	001320		CMP	@MSIZELO,RSIZE	:SEE IF SIZE JUMPERS AGREE WITH MEMORY TOP
4217	022030	001407				BEQ	19\$:BRANCH IF TOP OF MEMORY AGREES WITH
4218								:THE SIZE JUMPERS
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

:TEST 41 PARITY REPORTING THRU THE MAP, MAIN MEMORY EVEN WORD
:*****
: THIS TEST FORCES A PARITY ERROR IN MAIN MEMORY AT PHYSICAL
: ADDRESS 00040000 BY SETTING ONE BIT IN EACH BYTE OF THE
: WORD. THE TEST THEN FORCES THE EVEN WORD PARITY BITS TO
: ONES AND READS ADDRESS 00040000 THRU THE UNIBUS MAP. THEN
: IT CHECKS TO SEE THAT THE PARITY ERROR 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.
:*****

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 1 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\$,SLPERR	: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,CONTRL		: 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 @#CONTRL,PCONTR		: SAVE CONTROL REGISTER FOR TYPE OUT
4280	022306	012737	030000	001300		MOV #30000,PMANT		: SAVE DATA IN MAINTENANCE REGISTER
4281	022314	000002				RTI		: RETURN TO TEST AND CHECK PARITY
4282								
4283								
4284								
4285								
4286								
4287								
4288								
4289								
4290								
4291								
4292								
4293								
4294								
4295								
4296								
4297								
4298								
4299								
4300								

```

*****
: TEST 42 PARITY REPORTING THRU THE MAP, MAIN MEMORY ODD WORD
:
: THIS TEST FORCES A PARITY ERROR IN MAIN MEMORY AT PHYSICAL
: ADDRESS 00040002 BY SETTING ONE BIT IN EACH BYTE OF THE
: WORD. THE TEST THEN FORCES THE ODD WORD PARITY BITS TO
: ONES AND READS ADDRESS 00040002 THRU THE UNIBUS MAP. THEN
: IT CHECKS TO SEE THAT THE PARITY ABORT OCCURRED AND THAT THE
: CONDITION WAS CORRECT BEFORE GOING TO THE NEXT TEST.
:
: ERRORS ARE REPORTED INLINE IN THE TEST CODE, AND THE PARITY
: VECTOR WILL TRAP TO LABEL 10$ AT THE END OF THIS TEST.
:
: THIS TEST IS BEING RUN WITH ALL MEMORY REFERENCES GOING THRU
: THE UNIBUS MAP.
*****

```

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 @#CONTRL,\$TMP5		: SAVE CONTROL REG TO RESTORE CACHE
4309	022354	012737	000014	177746		MOV #14,@#CONTRL		: 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,@#LREG1		: LOWEST MAP REGISTER POINTS TO SR
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 EROR 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	01102				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	@MEMERR, 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, @MAINT	:SAVE DATA IN MAINTENANCE REGISTER
4341	022566	000002				RTI		:RETURN TO TEST AND CHECK PARITY

 :TEST 43 PARITY REPORTING THRU THE MAP, CACHE DATA GROUP 0

THIS TEST FORCES A PARITY ERROR IN CACHE GROUP 0 AT PHYSICAL ADDRESS 00040000 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 00040000 THRU THE UNIBUS MAP. THEN IT CHECKS TO SEE THAT THE PARITY ABORT OCCURRED AND THAT THE CONDITION WAS CORRECT BEFORE GOING TO THE NEXT TEST.

ERRORS ARE REPORTED INLINE IN THE TEST CODE, AND THE PARITY VECTOR WILL TRAP TO LABEL 10\$ AT THE END OF THIS TEST.

THIS TEST IS BEING RUN WITH ALL MEMORY REFERENCES GOING THRU THE UNIBUS MAP.

 :TST43:

4362	022570					MOV	#TST44, NXTTST	:SAVE STARTING ADDRESS OF NEXT TEST
4363	022570	000004						:FOR ESCAPE ON PARITY ERRORS
4364	022572	012737	023056	001324		MOV	#-1, 21\$:RESTORE NEG ONE FOR NEXT ITERATION
4365								:SEE IF GROUP 0 IS DISABLED
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	177747	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\$,SLPERR	: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	@PLOADRS,PLOADR	:SAVE LOWER CACHE ADDR REG
4411	023024	013737	177742	001272		MOV	@PHIADRS,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,PMMAINT	:SAVE DATA IN MAINTENANCE REGISTER
4415	023054	000002				RTI		:RETURN TO TEST AND CHECK PARITY
4416								
4417								
4418								
4419								
4420								
4421								
4422								
4423								
4424								
4425								
4426								
4427								
4428								

```

*****
:TEST 44 PARITY REPORTING THRU THE MAP, CACHE DATA GROUP 1
:
: THIS TEST FORCES A PARITY ERROR IN CACHE GROUP 1 AT PHYSICAL
: ADDRESS 00040000 BY SETTING BIT 15 IN THE WORD. THE TEST
: THEN FORCES GROUP 1 LOW BYTE PARITY BIT TO ZERO AND EVEN
: WORD HIGH BYTE PARITY BIT TO ONE BEFORE READING ADDRESS 00040000
: THRU THE UNIBUS MAP. THEN IT CHECKS TO SEE THAT THE PARITY
: ABORT OCCURRED AND THAT THE CONDITION WAS CORRECT BEFORE
: GOING TO THE NEXT TEST.
:
:

```

```

4429
4430
4431
4432
4433
4434
4435
4436 023056 000004
4437 023060 012737 023344 001324
4438
4439 023066 012737 177777 023232
4440 023074 032737 000010 177746
4441 023102 001120
4442 023104 013737 177746 001202
4443 023112 012737 023604 001200
4444 023120 005037 001274
4445 023124 012737 177777 177744
4446 023132 013737 001240 172354
4447 023140 005077 156102
4448 023144 012777 040000 156072
4449 023152 012705 177750
4450 023156 012704 140000
4451 023162 012714 100000
4452
4453
4454
4455 023166 012737 000044 177746
4456 023174 011401
4457 023176 005001
4458 023200 012737 023304 000114
4459 023206 010137 177750
4460 023212 000241
4461
4462 023214 011401
4463 023216 011503
4464 023220 105015
4465 023222 006701
4466 023224 012701 020100
4467
4468
4469 023230 005227
4470 023232 177777
4471 023234 001764
4472 023236 023737 001200 001274
4473 023244 001402
4474 023246 010302
4475 023250 104051
4476 023252 012737 177777 177744
4477 023260 012737 005764 000114
4478 023266 013737 001202 177746
4479 023274 012737 023066 001110
4480 023302 000420
4481
4482
4483 023304 013737 177740 001270
4484 023312 013737 177742 001272
  
```

```

: * 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.
: *
: *****
TST44: SCOPE
MOV #TST45,NXTTST ;SET UP ESCAPE POINTER IN CASE OF
;RANDOM PARITY ABORTS
20$: MOV #-1,21$ ;RESTORE NEG ONE FOR NEXT ITERATION
BIT #BIT3,CONTRL ;SEE IF GROUP 1 IS DISABLED
BNE TST45 ;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,@LREGU ;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 P.B. TO BE OFF
;FORCE SELECTION OF GROUP 0 ON READ
;PUT (40000) & (40002) IN GROUP 1
MOV #44,@CONTRL ;CLEAR R1 FOR MOMENT
MOV (R4),R1 ;SET PARITY VECTOR TO 10$
CLR R1 ;FORCE GP1 PARITY BITS TO 0 ON 2ND PASS
MOV #10$,CACHVEC ;THIS IS A SYNC POINT FOR SCOPING
MOV R1,@MAINT ;THE LOW BYTE HAS ITS P.B. OFF
CLC ;DATA FETCH SHOULD CAUSE PARITY ABORT
;SAVE MAINT REG IN CASE NO TRAP
;CLEAR MAINT REG IN CASE NO TRAP
;DUPPY INST WITH P.B.'S OFF
;FORCE GROUP 1 LOW BYTE PARITY BIT
;TO 0 AND MAIN MEMORY EVEN WORD HIGH
;BYTE PARITY BIT TO 1
;INCREMENT NEXT WORD TO 0
4462: MOV (R4),R1
MOV (R5),R3
CLRB (R5)
SXT R1
MOV #020100,R1
;BRANCH ON FIRST PASS ONLY
;SEE IF ERROR CONDITION MATCHES EXPECTED
;BRANCH IF ERROR WAS EXPECTED ONE
;R2 HOLDS MAINT REG FOR ERROR TYPE OUT
;FAILURE IN PARITY REPORTING
;CLEAR MEMORY ERROR REGISTER
;RESTORE PARITY SERVICE ROUTINE
;RESTORE CACHE TO FORMER CONDITION
;SET LOOP POINTER TO START OF TEST
;TEST OVER GO TO NEXT TEST
21$: INC (PC)+
;WORD
; -1
BEQ 1$
CMP $TMP4,PPARER
BEQ 2$
MOV R3,R2
ERROR 51
MOV #-1,MEMERR
MOV @MEMER,CACHVEC
MOV $TMP5,CONTRL
MOV #7,$LPERR
BR 1$
2$: MOV @LOADRS,PLOADR ;SAVE LOWER CACHE ADDR REG
MOV @HIADRS,PHIADR ;SAVE HIGH BITS OF FAILING ADDR
10$:
  
```

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,PMANT ;SAVE DATA IN MAINTENANCE REGISTER
4488 023342 000002 RTI ;RETURN TO TEST AND CHECK PARITY
4489
4490
4491
4492
4493

*TEST 45 CHECK BYP BITS IN UBMR
* THIS TEST IS EXECUTED ONLY ON KB11-EM OR 11/74 (KB11CM)

4495
4496 023344 000004
4497 023346 012737 023500 001324 TST45: SCOPE
4498 023354 105737 001331 MOV #TST46,NXTTST
4499 023360 001003 TSTB KB11EM ;IS THIS A KB11-EM?
4500 023362 105737 001332 BNE 10\$;BR IF YES
4501 023366 001444 TSTB KB11CM ;IS THIS A MODIFIED PROCESSOR?
4502 023370 012700 170202 BEQ 5\$
4503 023374 005010 10\$: MOV #170202,R0
4504 023376 032710 100000 4\$: CLR (R0)
4505 023402 001405 BIT #BYP,(R0)
4506 023404 010037 001154 BEQ 1\$
4507 023410 011037 001156 MOV R0,\$REG0
4508 023414 104053 MOV (R0),\$REG1
4509 ERROR 53
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

*TEST 46 CHECK FOR PRESENCE OF MKA11 REGS. ON KB11-EM OR KB11CM
* THIS TEST IS EXECUTED ONLY ON KB11-EM OR 11/74 (KB11CM)

4530
4531
4532
4533
4534
4535 023500 000004
4536 023502 012737 023640 001324 TST46: SCOPE
4537 172100 MOV #TST47,NXTTST
4538 MSER= 177744
4539 023510 177744 TSTB KB11EM ;KB11-EM?
4540 023510 105737 001331

```

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 SOME MORE
4552 023556 000426          BR     98$         ;TEST IS OVER.
4553
4554 023560 012705 001000   :***** TRAP HANDLER ***
4555 023564 077501          66$:    MOV     #1000, R5 ;SET COUNTER AND...
4556 023566 005737 177744   SOB     R5          ;WAIT A WHILE FOR THE GATES TO SETTLE
4557 023572 001004          TST    @#MSER      ;ANY BITS SET IN MEM. ERR. REG?
4558 023574 032737 000020 177766   BNE    68$         ;YES, ERROR
4559 023602 001011          BIT    #BIT4, @#CPUERR ;UNIBUS ERROR BIT SET?
4560 023604 010237 001160   BNE    77$         ;YES, OK
4561 023610 013737 177744 001172   68$:    MOV     R2, $REG2 ;PRINT OFFENDING CSR
4562 023616 013737 177766 001174   MOV     @#MSER, $TMP1 ;MSER
4563 023624 104055          MOV     @#CPUERR, $TMP2 ;CPUERR
4564 023626 005037 177766   77$:    ERROR    55 ;REPORT ERROR
4565 023632 000002          CLR    @#CPUERR   ;CLEAR THE ERROR REGISTER
4566
4567
4568 023634 012637 000004   98$:    MOV     (SP)+, @#4 ;RESTORE CPU ERROR VECTOR
4569
4570 023640
4571
4572
4573
4574
4575
4576
4577 023640 000004          99$:
4578 023642 012737 000005 001204   :*****
4579
4580 023650 013746 177776   :TEST 47 CHECK CACHE BYPASS ON UNIBUS MAP PAGE
4581 023654 042716 000020   :THIS TEST CHECKS CACHE BYPASS ON UNIBUS-MAP-PAGE
4582 023660 012746 023666   :THIS TEST IS EXECUTED ONLY ON KB11-EM OR 11/74 (KB11CM)
4583 023664 000002   :THIS TEST USES MEMORY MANAGEMENT
4584 023666
4585
4586 023666 012737 024360 001324   :*****
4587 023674 105737 001331   TST47: SCOPE
4588 023700 001005          MOV     #5, $TIMES ;DO 5 ITERATIONS
4589 023702 105737 001332   MOV     @#PS, -(SP)
4590 023706 001002          BIC    #20, (SP)   ;CLEAR T BIT IF SET
4591 023710 000137 024360   MOV     #1$, -(SP)
4592 023714 012737 170000 172354 1$:
4593 023722 012737 010406 172314   RTI
4594
4595 023730 012705 170200   MOV     #SEOP, $NEXTTST ;IS THIS A KB11-EM?
4596
4597
4598
4599
4600
4601
4602
4603
4604
4605
4606
4607
4608
4609
4610
4611
4612
4613
4614
4615
4616
4617
4618
4619
4620
4621
4622
4623
4624
4625
4626
4627
4628
4629
4630
4631
4632
4633
4634
4635
4636
4637
4638
4639
4640
4641
4642
4643
4644
4645
4646
4647
4648
4649
4650
4651
4652
4653
4654
4655
4656
4657
4658
4659
4660
4661
4662
4663
4664
4665
4666
4667
4668
4669
4670
4671
4672
4673
4674
4675
4676
4677
4678
4679
4680
4681
4682
4683
4684
4685
4686
4687
4688
4689
4690
4691
4692
4693
4694
4695
4696
4697
4698
4699
4700
4701
4702
4703
4704
4705
4706
4707
4708
4709
4710
4711
4712
4713
4714
4715
4716
4717
4718
4719
4720
4721
4722
4723
4724
4725
4726
4727
4728
4729
4730
4731
4732
4733
4734
4735
4736
4737
4738
4739
4740
4741
4742
4743
4744
4745
4746
4747
4748
4749
4750
4751
4752
4753
4754
4755
4756
4757
4758
4759
4760
4761
4762
4763
4764
4765
4766
4767
4768
4769
4770
4771
4772
4773
4774
4775
4776
4777
4778
4779
4780
4781
4782
4783
4784
4785
4786
4787
4788
4789
4790
4791
4792
4793
4794
4795
4796
4797
4798
4799
4800
4801
4802
4803
4804
4805
4806
4807
4808
4809
4810
4811
4812
4813
4814
4815
4816
4817
4818
4819
4820
4821
4822
4823
4824
4825
4826
4827
4828
4829
4830
4831
4832
4833
4834
4835
4836
4837
4838
4839
4840
4841
4842
4843
4844
4845
4846
4847
4848
4849
4850
4851
4852
4853
4854
4855
4856
4857
4858
4859
4860
4861
4862
4863
4864
4865
4866
4867
4868
4869
4870
4871
4872
4873
4874
4875
4876
4877
4878
4879
4880
4881
4882
4883
4884
4885
4886
4887
4888
4889
4890
4891
4892
4893
4894
4895
4896
4897
4898
4899
4900
4901
4902
4903
4904
4905
4906
4907
4908
4909
4910
4911
4912
4913
4914
4915
4916
4917
4918
4919
4920
4921
4922
4923
4924
4925
4926
4927
4928
4929
4930
4931
4932
4933
4934
4935
4936
4937
4938
4939
4940
4941
4942
4943
4944
4945
4946
4947
4948
4949
4950
4951
4952
4953
4954
4955
4956
4957
4958
4959
4960
4961
4962
4963
4964
4965
4966
4967
4968
4969
4970
4971
4972
4973
4974
4975
4976
4977
4978
4979
4980
4981
4982
4983
4984
4985
4986
4987
4988
4989
4990
4991
4992
4993
4994
4995
4996
4997
4998
4999
5000

```

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 #SOMOM1,@#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,@#PPR3	:ENABLE 22-BIT MAPPING
4636	024150	012737	000001	177572	MOV #1,@#PPR0	: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 (H1) 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$: TST      (R1)          ;REFERENCE TEST DATA THROUGH  
4659                                     ;THE UNIBUS MAP (WITH CACHE  
4660                                     ;BYPASS SET) AND CHECK  
4661                                     ;IF THE REFERENCE WAS A MISS  
4662  
4663 024234 062701 000002          6$: 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$: TST      (R0)          ;REFERENCE TEST-DATA AGAIN  
4672                                     ;AND CHECK THAT IT WAS  
4673                                     ;INVALIDATED PREVIOUSLY  
4674 024270 032737 000010 177752          BIT      #10,@HITMIS  
4675 024276 001413          BEQ      8$  
4676 024300 013737 177746 001154          MOV      @CONTRL,SREG0  
4677 024306 010537 001156          MOV      R5,SREG1  
4678 024312 162737 000002 001156          SUB      #2,SREG1  
4679 024320 010037 001160          MOV      R0,SREG2  
4680 024324 104057          ERROR    57  
4681                                     ;ADDRESS OF UB MAP REGISTER THAT  
4682                                     ;IS BEING TESTED  
4683                                     ;TEST-DATA ADDRESS  
4684                                     ;TEST DATA REFERENCE WAS NOT  
4685                                     ;A MISS. TEST-DATA WAS  
4686                                     ;PREVIOUSLY REFERENCED IN  
4687                                     ;BYPASS (ON UBMR) MODE, HENCE  
4688                                     ;IT SHOULD HAVE BEEN  
4689                                     ;INVALIDATED. BUT, SUBSEQUENT  
4690                                     ;REFERENCE WAS NOT A MISS,  
4691                                     ;AS EXPECTED  
4692  
4693  
4694 024326 062700 000004          8$: ADD      #4,R0  
4695 024332 077223          SOB      R2,7$          ;DONE?  
4696 024334 020527 170366          CMP      R5,@MAPH35  
4697 024340 001407          BEQ      10$          ;DONE CHECKING ALL 31 UBMR'S  
4698  
4699  
4700  
4701  
4702  
4703  
4704  
4705  
4706  
4707  
4708  
172354  
10$:  
;:*****  
.SBTTL  END OF PASS ROUTINE  
;*INCREMENT THE PASS NUMBER ($PASS)  
;*INDICATE END-OF-PROGRAM AFTER 1 PASSES THRU THE PROGRAM  
;*TYPE 'END PASS #XXXXX TOTAL NUMBER OF ERRORS SINCE LAST REPORT YYYYY'  
;*WHERE XXXXX AND YYYYY ARE DECIMAL NUMBERS  
;*IF SW12=1 INHIBIT TRACE TRAP  
;*IF THERES A MONITOR GO TO IT
```



```

4709
4710
4711 024360
4712 024360 000004
4713 024362 005037 177572
4714 024366 005037 172516
4715 024372 10442C
4716 024374 005037 001102
471 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
CLR MMR0
CLR MMR3
TBITR
CLR $STNM
CLR $TIMES
INC $PASS
BIC #100000,$PASS
DEC (PC)+
$EOPCT: .WORD 1
BGT $DOAGN
MOV (PC)+,@(PC)+
$ENDCT: .WORD 1
$EOPCT
TYPE .65$
BR 64$
::65$: .ASCIZ <12><15>/END PASS #/
64$: MOV $PASS,-(SP)
TYPDS
TYPE .67$
BR 66$
::67$: .ASCIZ / TOTAL ERRORS SINCE LAST REPORT /
66$: MOV $ERTL,-(SP)
TYPDS
TYPE .SCLF
CLR $ERTL
$GET42: MOV @#42,R0
BEQ $DOAGN
CLR -(SP)
MOV #SCLR.T,-(SP)
BR $RTN
SCLR.T: MOV @#42,R0
BEQ $DOAGN
RESET
$ENDAD: JSR PC,(R0)
NOP
NOP
NOP
$DOAGN: MOV @#PS,-(SP)
BIC #20,(SP)
BIT #BIT12,@#SWR
BNE 1$
COM $TBIT
BMI 1$
BIS #20,(SP)
1$: MOV #SLOOP,-(SP)

```

```

: LOOP ON LAST TEST
: TURN OFF FULL RELOCATION
: DISABLE THE UNIBUS MAP
: RESTORE THE T BIT IF IT WAS ON
: ZERO THE TEST NUMBER
: ZERO THE NUMBER OF ITERATIONS
: INCREMENT THE PASS NUMBER
: DON'T ALLOW A NEG. NUMBER
: LOOP?
:: YES
:: RESTORE COUNTER
:: TYPE ASCIZ STRING
:: GET OVER THE ASCIZ
:: SAVE $PASS FOR TYPEOUT
:: TYPE PASS NUMBER
:: GO TYPE--DECIMAL ASCII WITH SIGN
:: TYPE ASCIZ STRING
:: GET OVER THE ASCIZ
:: SAVE $ERTL FOR TYPEOUT
:: TOTAL NUMBER OF ERRORS
:: GO TYPE--DECIMAL ASCII WITH SIGN
:: TYPE CARRIAGE RETURN, LINE FEED
:: CLEAR ERROR TOTAL
:: GET MONITOR ADDRESS
:: BRANCH IF NO MONITOR
:: INSURE THE 'T' BIT IS CLEAR
:: SETUP FOR AN RTI OR RTT
:: GO DO AN RTI OR RTT TO LOAD THE PSW
:: WITH A CLEARED 'T' BIT
:: INSURE R0 CONTAINS THE MONITORS
:: RETURN ADDRESS
:: CLEAR THE WORLD
:: GO TO MONITOR
:: SAVE ROOM
:: FOR
:: ACT11
:: PUT THE PS ON THE STACK AND
:: CLEAR THE 'T' BIT
:: RUN WITH TRACE TRAP?
:: BR IF NO
:: IS IT TIME FOR TRACE TRAP
:: BR IF NO
:: SET TRACE TRAP
:: JUMP TO START OF TEST

```


4765 024646 000002
4766
4767
4768 024650
4769 024650 000137 010652
4770 024654 000000
4771 024656 377 377 000
4772 024662
4773
4774
4775 024662 041600 052520 052440
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
4782 024726 046505 000200
4783 024732 041113 030461 041055
4784 024740 041457 000200
4785 024744 041113 030461 041455
4786 024752 020115 020040 020040
4787 024760 020040 020040 020040
4788 024766 020040 020040 100040
4789 024774 000
4790 024775 113 030502 026461
4791 025002 100105 000
4792
4793
4794
4795 025005 116 052117 052040
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
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
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

\$RTRN: RTI ;:RETURN--THIS IS CHANGED TO
;:AN 'RTI' IF 'RTI' IS A LEGAL
;:INSTRUCTION
\$LOOP: ;:RETURN
\$TBIT: JMP @#LOOP ;:'T' BIT STATE INDICATOR
;:WORD 0 ;:NULL CHARACTER STRING
\$ENULL: .BYTE -1,-1,0
;:EVEN

:MESSAGES
MSG1: .ASCIZ<CRLF> 'CPU UNDER TEST FOUND TO BE A ''

MSG2: .ASCIZ 'KB11-EM'<CRLF>

MSG3: .ASCIZ 'KB11-B/C'<CRLF>

MSG4: .ASCIZ 'KB11-CM ''<CRLF>

MSG5: .ASCIZ 'KB11-E'<CRLF>

.SBTTL ERROR MESSAGES AND DATA TABLES
EM1: .ASCIZ ?NOT THE CORRECT TRAP CONDITION THRU ERRVEC (#004)?

EM2: .ASCIZ ?UNEXPECTED CPU TRAP THRU ERRVEC (#004)?

EM3: .ASCIZ ?UNEXPECTED CACHE PARITY ERROR THRU CACHVEC. WILL RETRY TEST ONCE?

4821	025232	047117	042503	000
4822	025237	125	042516	050130
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
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
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
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
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
4874	025714	054522	047440	020106
4875	025722	040515	020120	042522
4876	025730	044507	052123	051105

EM4: .ASCIZ ?UNEXPECTED MAIN MEMORY PARITY ERROR THRU CACHVEC, WILL RETRY TEST ONCE?

EM5: .ASCIZ ?UNEXPECTED MEMORY MANAGEMENT TRAP, MEMORY MANAGEMENT STATUS REGISTERS?

EM6: .ASCIZ ?SUMMARY OF MAP REGISTERS THAT TIMED OUT ON READ?

EM7: .ASCIZ ?SUMMARY OF CACHE REGISTERS THAT TIMED OUT ON READ?

EM10: .ASCIZ ?SUMMARY OF MAP REGISTERS NOT HOLDING ZERO IN LOW 16 BITS?

EM11: .ASCIZ ?SUMMARY OF MAP REGISTERS NOT HOLDING ZERO IN UPPER 6 BITS?

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
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
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
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
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
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

EM12: .ASCIZ ?POSSIBLE ERROR IN MAP REGISTER DATA PATH (MAP REG 00)?

EM13: .ASCIZ ?NOW PROBABLE ERROR IN MAP REGISTER DATA PATH (MAP REG 20)?

EM14: .ASCIZ ?SUMMARY OF DUAL ADDRESSING ERRORS ON LOADING MAP REGISTERS?

EM15: .ASCIZ ?SUMMARY OF COUNT PATTERN FAILURES IN LOWER 16 BITS OF MAP REGISTERS?

EM16: .ASCIZ ?SUMMARY OF COUNT PATTERN FAILURES IN UPPER 6 BITS OF MAP REGISTERS?

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 MAINTENANCE 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
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
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
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
5027	027444	047105	02503	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
5039	027552	041516	042105	041110
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

.ASCII ?POSSIBLE ERROR IN CACHE REGISTER DATA PATH?

EM23: .ASCII ?SUMMARY OF COUNT PATTERN FAILURES IN CACHE CONTROL REGISTER?

EM24: .ASCII ?SUMMARY OF COUNT PATTERN FAILURES IN CACHE MAINTENANCE REGISTER?

EM25: .ASCII ?REFERENCED MAP REGISTER 0 WITH ADDRESS ONE BIT DIFFERENT THAN 770200?

EM26: .ASCII ?REFERENCED CACHE LOW ADDRESS REGISTER WITH ADDRESS ONE BIT?<CRLF>

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
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
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
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
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
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
5155	030762	041440	051125	042522
5156	030770	052116	051440	052105

EM34: .ASCII ?THE NUMBER OF MAP REGISTERS REMOVED BY JUMPER SETTING DOES?<CRLF>

.ASCII ?NOT AGREE WITH THE NUMBER FOUND TO BE MISSING?

EM35: .ASCII ?THE SIZE JUMPERS ON THE UNIBUS MAP ARE NOT SET IN?<CRLF>

.ASCII ?THEIR DEFAULT POSITION, WHICH ALLOWS UNIBUS ADDRESSES?<CRLF>

.ASCII ?000000 TO 757776 TO REFERENCE MAIN MEMORY?<CRLF>

.ASCII ?THEIR CURRENT SETTING ALLOWS ONLY:?

5157	030776	044524	043516	040440
5158	031004	046114	053517	020123
5159	031012	047117	054514	000072
5160	031020	040515	020120	042522
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
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
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
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
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

EM36: .ASCIZ ?MAP REGISTER UNDER TEST DID NOT RESPOND IN DUAL MAPPING TEST?

EM37: .ASCIZ ?SUMMARY OF UNIBUS ADDRESS ERRORS, WITH THE MAP RELOCATION DISABLED?

EM40: .ASCIZ ?MAIN MEMORY TIMEOUT OVER THE UNIBUS DID NOT OCCUR PROPERLY?

EM41: .ASCIZ ?RELOCATION THRU THE MAP WAS NOT CORRECT, FULL ADD?

EM42: .ASCIZ ?RELOCATION THRU THE MAP WAS NOT CORRECT, CARRY PROPAGATION?

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	043111
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
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
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
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
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
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
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

EM44: .ASCIZ ?PARITY REPORTING THRU THE MAP IS NOT CORRECT?

EM45: .ASCII ?MAIN MEMORY TIMEOUT OVER THE UNIBUS DID NOT OCCUR PROPERLY?<CRLF>

.ASCIZ ?TEST CODE BEING RUN OVER UNIBUS?

EM46: .ASCII ?RELOCATION THRU THE MAP WAS NOT CORRECT, FULL ADD?<CRLF>

.ASCIZ ?TEST CODE BEING RUN OVER UNIBUS?

EM47: .ASCII ?RELOCATION THRU THE MAP WAS NOT CORRECT, CARRY PROPAGATION?<CRLF>

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?<CR LF>
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	044503	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?<CR LF>
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 MK11 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
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
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
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
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
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
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

EM56: .ASCIZ ?TEST DATA REFERENCE NOT A HIT?

EM57: .ASCII ?TEST DATA REFERENCE NOT A MISS?<15><12>

.ASCIZ ?CACHE BYPASS ON UNIBUS MAP PAGE DID NOT INVALIDATE CACHED DATA?

EM201: .ASCIZ ?THE FOLLOWING REGISTERS TIMED OUT WHEN READ?

EM202: .ASCIZ ?THE FOLLOWING MAP REGISTERS WILL NOT CLEAR?

EM203: .ASCIZ ?THE FOLLOWING ARE DUAL ADDRESSING ERRORS IN THE UNIBUS MAP?

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 REFERENC'D 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		.ASCIIZ ?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		.ASCIIZ ? 'DR' '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		.ASCIIZ ? 'DR' 'AND' 'DR' '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		.ASCIIZ ?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	DH15: .ASCII ?MAPREG MAPREG EXPECTD EXPECTD RECEIVD RECEIVD?<CRLF>
5504	034600	040515	051120	043505	
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?

Line	Code	Address	Value	Value	Value	Message
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 ?EXPECTED 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 REFERENC'D 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 (PUERR)/
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		.ASCIIZ	?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:	.ASCIIZ	?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:	.ASCIIZ	?EXPECTD RECEIVD	ADDRSLOAD	TESTNO ERRORPC?
5683	036540	020104	042522	042503					
5684	036546	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		.ASCIIZ	?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		.ASCIIZ	?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,\$TMPO,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							
5854	040020	000017			SAVEA:	.EVEN .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	

DM36	035342	789	5568#				
DM40	035434	802	837	5578#			
DM41	035514	809	845	5587#			
DM42	035574	816	853	5596#			
DM43	035674	823	861	5607#			
DM44	035734	829	869	5613#			
DM5	034057	612	5442#				
DM53	036100	884	891	5630#			
DM55	036127	897	5634#				
DM56	036172	904	5640#				
DM57	036237	911	5647#				
DM6	034163	619	626	5455#			
DISPLA=	177570	39#	1037*	1068*	2927*	3136*	3940*
DT1	036764	585	5712#				
DT10	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#				

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	001332	560#	2104*	2152*	2164	2486	3627	4221	4500	4542	4589			
KB11E	001330	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

.SCATC	1#	422
.SCMTA	1#	460
.SDB2D	1#	
.SDB2O	1#	1630
.SDIV	1#	
.SEOP	1#	4699
.SERRO	1#	1041
.SERRT	1#	
.SMULT	1#	
.SPOWE	1#	1583
.SRAND	1#	
.SRDDE	1#	
.SRDOC	1#	
.SREAD	1#	
.SSAVE	1#	1283
.SSB2D	1#	
.SSB2O	1#	
.SSCOP	1#	966
.SSIZE	1#	
.SSUPR	1#	
.STRAP	1#	1548
.STYPB	1#	
.STYPD	1#	1480
.STYPE	1#	1329
.STYPO	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)