

11/70

UNIBUS MAP DIAGNOSTIC  
MD-11-DEKBF-B

EP DEKBF B DL A

COPYRIGHT 1976

FICHE 1 OF 1

NOV 1976

digital

MADE IN USA

This section contains a grid of 100 small diagnostic maps, arranged in 10 rows and 10 columns. Each map displays a different component or configuration of the UNIBUS system, with various data points and labels. The maps are organized into a structured grid, with each cell containing a unique diagnostic view. The content is dense and technical, typical of a diagnostic manual for a complex system like the UNIBUS.

A small, isolated diagnostic map located in the bottom right corner of the page. It contains a grid of data points and labels, similar in format to the larger maps in the main grid.



801

DDP-11 TO UNIBUS MAP DIAGNOSTIC MACY11 27(732) 20-SEP-76 13:32 PAGE 1  
DENSFB.P11

.REM %

CO1

.MAIN. MACY11 27(657)  
DEKBF8.MAN

PAGE 1

.REM 2

IDENTIFICATION  
-----

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

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.

THE SOFTWARE DESCRIBED IN THIS DOCUMENT IS FURNISHED TO THE PURCHASER UNDER A LICENSE FOR USE ON A SINGLE COMPUTER SYSTEM AND CAN BE COPIED (WITH INCLUSION OF DIGITAL'S COPYRIGHT NOTICE) ONLY FOR USE IN SUCH SYSTEM, EXCEPT AS MAY OTHERWISE BE PROVIDED IN WRITING BY DIGITAL.

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 BY DIGITAL EQUIPMENT CORPORATION

UNIBUS MAP DIAGNOSTIC MACY11 27(657) DEKBF8.MAN





100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155

1. ABSTRACT  
-----

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

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

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

(SEE SECTION 6.3 FOR AN EXAMPLE OF THE ERROR TYPEOUTS.)

THE PROGRAM LOADS "044" INTO THE MICRO BREAK REGISTER (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.

2. REQUIREMENTS  
-----

2.1 EQUIPMENT

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

2.2 STORAGE

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

FO1

156

FROM THE UNIBUS.

GO1

157  
158  
159  
160  
161  
162  
163  
164  
165  
166

.MAIN. MACY11 27(657)  
DEKBFA.MAN

PAGE 4

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.



167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212

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

213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268

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

269  
270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324

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

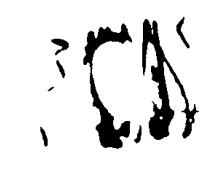
IF SWITCH 9 IS UP, THE PROGRAM WILL LOOP BACK TO THE POINT WHERE THE INSTRUCTION THAT CAUSED THE ERROR WAS EXECUTED, WITHOUT ALLOWING ANY OF THE CONDITIONS TO CHANGE. THIS WILL PROVIDE THE TIGHTEST POSSIBLE SCOPE LOOP. IF SWITCH 9 IS DOWN, EACH ERROR WILL BE REPORTED AND LOGGED AND, AT THE END OF EACH 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





K01

PDP-11/70 UNIBUS MAP DIAGNOSTIC MACY11 27(732) 20-SEP-76 13:32 PAGE 10  
DEKBF8.P11

325

170312 000001 015232

1/2

.MAIN. MACY11 27(657)  
DEKBFA.MAN

PAGE 8

326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359

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

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

- 7. RESTRICTIONS  
-----
- 7.1 STARTING RESTRICTIONS  
NONE
- 7.2 OPERATING RESTRICTIONS  
NONE

360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415

.MAIN. MACY11 27(657)  
DEKBFA.MAN

PAGE 9

8. MISCELLANEOUS

8.1 EXECUTION TIME

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

8.2 ADDRESS GENERATION IN THE PDP-11/70

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

A. VIRTUAL ADDRESS	15 14 13 12 11 10 09 08 07 06 05 04 03
A1. PAGE NUMBER (0-7)	15 14 13
A2. OFFSET	12 11 10 09 08 07 06 05 04 03
B. P.A.R. [PAGE NO.] +	15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00
C. PHYSICAL ADDR.	21 20 19 18 17 16 15 14 13 12 11 10 09 08 07 06 05 04 03
C1. 17XXXXXX=> U.B.ADR.	21 20 19 18
C2. MAPPING REG.NO. (0-36)	17 16 15 14 13
C3. OFFSET	12 11 10 09 08 07 06 05 04 03
D. MAP REG. [NO.] +	21 20 19 18 17 16 15 14 13 12 11 10 09 08 07 06 05 04 03
E. PHYSICAL ADDR.	21 20 19 18 17 16 15 14 13 12 11 10 09 08 07 06 05 04 03

DESCRIPTION OF LINES:

A: VIRTUAL ADDRESS (16 BITS)

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

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

C: PHYSICAL ADDRESS CREATED BY MEMORY MANAGEMENT. (22 BITS)

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

D: MAP REGISTER (22 BITS). ADDED TO BITS <2:00> OF UNIBUS ADDRE



NO1

PDP-11/70 UNIBUS MAP DIAGNOSTIC MACY11 27(732) 20-SEP-76 13:32 PAGE 13  
DEKBFB.P11

416

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

802

PDF-11 TO UNIBUS MAP DIAGNOSTIC MACY11 27(732) 20-SEP-76 13:32 PAGE 14  
DEKBF8.P11

417  
418  
419  
420  
421

.MAIN. MACY11 27(657)  
DEKBF8.MAN

PAGE 10

PCP-11  
UNIBUS  
MAP  
DIAGNOSTIC  
MACY11  
27(732)  
20-SEP-76  
13:32  
PAGE 15  
DEKBFB.P11

.MAIN. MACY11 27(657)  
DEKBFA.MAN

PAGE 11

9. PROGRAM DESCRIPTION  
-----

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

a  
.END

*Handwritten mark*

404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489

```

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

.SBTTL OPERATIONAL SWITCH SETTINGS
:*
:*      SWITCH          USE
:*      -----          -
:*      15             HALT ON ERROR
:*      14             LOOP ON TEST
:*      13             INHIBIT ERROR TYPEOUTS
:*      12             INHIBIT TRACE TRAP
:*      11             INHIBIT ITERATIONS
:*      10             BELL ON ERROR
:*      9              LOOP ON ERROR
:*      8              LOOP ON TEST IN SWR(6:0)
:*      7              INHIBIT MULTIPLE ERROR TYPEOUTS

.SBTTL BASIC DEFINITIONS

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

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

:*GENERAL PURPOSE REGISTER DEFINITIONS
R0= %0              ;;GENERAL REGISTER
R1= %1              ;;GENERAL REGISTER
R2= %2              ;;GENERAL REGISTER
R3= %3              ;;GENERAL REGISTER
R4= %4              ;;GENERAL REGISTER
R5= %5              ;;GENERAL REGISTER
R6= %6              ;;GENERAL REGISTER

```

001100  
001100  
000700  
000600  
  
177776  
  
177774  
177772  
177570  
177570  
  
000011  
000012  
000015  
000200  
  
000000  
000001  
000002  
000003  
000004  
000005  
000006

490 000007  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545

R7= %7  
.EQUIV R0,R10  
.EQUIV R1,R11  
.EQUIV R2,R12  
.EQUIV R3,R13  
.EQUIV R4,R14  
.EQUIV R5,R15  
.EQUIV R6,SP  
.EQUIV SP,KSP  
.EQUIV SP,SSP  
.EQUIV SP,USP  
.EQUIV R7,PC

:: GENERAL REGISTER  
:: GENERAL REGISTER  
:: GENERAL REGISTER  
:: GENERAL REGISTER  
:: GENERAL REGISTER  
:: GENERAL REGISTER  
:: GENERAL REGISTER  
:: STACK POINTER  
:: KERNEL STACK POINTER  
:: SUPERVISOR STACK POINTER  
:: USER STACK POINTER  
:: PROGRAM COUNTER

\*PRIORITY LEVEL DEFINITIONS

PRO= 0  
PR1= 40  
PR2= 100  
PR3= 140  
PR4= 200  
PR5= 240  
PR6= 300  
PR7= 340

:: PRIORITY LEVEL 0  
:: PRIORITY LEVEL 1  
:: PRIORITY LEVEL 2  
:: PRIORITY LEVEL 3  
:: PRIORITY LEVEL 4  
:: PRIORITY LEVEL 5  
:: PRIORITY LEVEL 6  
:: PRIORITY LEVEL 7

\*"SWITCH REGISTER" SWITCH DEFINITIONS

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

\*DATA BIT DEFINITIONS (BIT00 TO BIT15)

BIT15= 100000  
BIT14= 40000  
BIT13= 20000  
BIT12= 10000

000000  
000040  
000100  
000140  
000200  
000240  
000300  
000340  
  
100000  
040000  
020000  
010000  
004000  
002000  
001000  
000400  
000200  
000100  
000040  
000020  
000010  
000004  
000002  
000001  
  
100000  
040000  
020000  
010000





54  
55

546 004000  
 547 002000  
 548 001000  
 549 000400  
 550 000200  
 551 000100  
 552 000040  
 553 000020  
 554 000010  
 555 000004  
 556 000002  
 557 000001

BIT11= 4000  
 BIT10= 2000  
 BIT09= 1000  
 BIT08= 400  
 BIT07= 200  
 BIT06= 100  
 BIT05= 40  
 BIT04= 20  
 BIT03= 10  
 BIT02= 4  
 BIT01= 2  
 BIT00= 1  
 .EQUIV BIT09,BIT9  
 .EQUIV BIT08,BIT8  
 .EQUIV BIT07,BIT7  
 .EQUIV BIT06,BIT6  
 .EQUIV BIT05,BIT5  
 .EQUIV BIT04,BIT4  
 .EQUIV BIT03,BIT3  
 .EQUIV BIT02,BIT2  
 .EQUIV BIT01,BIT1  
 .EQUIV BIT00,BIT0

568  
 569  
 570 000004  
 571 000010  
 572 000014  
 573 000014  
 574 000014  
 575 000020  
 576 000024  
 577 000030  
 578 000034  
 579 000060  
 580 000064  
 581 000114  
 582 000240  
 583 000250

.\*BASIC "CPU" TRAP VECTOR ADDRESSES  
 ERRVEC= 4 ;: TIME OUT AND OTHER ERRORS  
 RESVEC= 10 ;: RESERVED AND ILLEGAL INSTRUCTIONS  
 TBITVEC=14 ;: "T" BIT  
 TRTVEC= 14 ;: TRACE TRAP  
 BPTVEC= 14 ;: BREAKPOINT TRAP (BPT)  
 IOTVEC= 20 ;: INPUT/OUTPUT TRAP (IOT) \*\*SCOPE\*\*  
 PWRVEC= 24 ;: POWER FAIL  
 EMTVEC= 30 ;: EMULATOR TRAP (EMT) \*\*ERROR\*\*  
 TRAPVEC=34 ;: "TRAP" TRAP  
 TKVEC= 60 ;: TTY KEYBOARD VECTOR  
 TPVEC= 64 ;: TTY PRINTER VECTOR  
 CACHVEC=114 ;: CACHE ERROR INTERRUPT VECTOR  
 PIRQVEC=240 ;: PROGRAM INTERRUPT REQUEST VECTOR  
 MMVEC= 250 ;: MEMORY MANAGEMENT VECTOR

584  
 585  
 586  
 587  
 588 177740  
 589 177742  
 590 177744  
 591 177746  
 592 177750  
 593 177752

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

594  
 595  
 596  
 597  
 598  
 599 177760  
 600  
 601 177762

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



602  
603  
604  
605  
606  
607  
608  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
650  
651  
652  
653  
654  
655  
656  
657

177764  
177766

SYSTID = 177764  
CPUERR = 177766

:: CURRENTLY ALL ZERO  
:: SYSTEM ID REGISTER  
:: CPU ERROR REGISTER HOLDS CONDITION THAT CAUSED  
:: THE TRAP TO ERRVEC (000004)

.SBTTL MEMORY MANAGEMENT DEFINITIONS

;\*MEMORY MANAGEMENT STATUS REGISTER ADDRESSES

:77572  
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

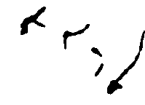
UDPDR0= 177620  
UDPDR1= 177622  
UDPDR2= 177624  
UDPDR3= 177626  
UDPDR4= 177630  
UDPDR5= 177632  
UDPDR6= 177634  
UDPDR7= 177636

;\*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



658		
659	177660	UDPAR0= 177660
660	177662	UDPAR1= 177662
661	177664	UDPAR2= 177664
662	177666	UDPAR3= 177666
663	177670	UDPAR4= 177670
664	177672	UDPAR5= 177672
665	177674	UDPAR6= 177674
666	177676	UDPAR7= 177676
667		
668		;*SUPERVISOR "I" PAGE DESCRIPTOR REGISTERS
669		
670	172200	SIPDR0= 172200
671	172202	SIPDR1= 172202
672	172204	SIPDR2= 172204
673	172206	SIPDR3= 172206
674	172210	SIPDR4= 172210
675	172212	SIPDR5= 172212
676	172214	SIPDR6= 172214
677	172216	SIPDR7= 172216
678		
679		;*SUPERVISOR "D" PAGE DESCRIPTOR REGISTERS
680		
681	172220	SDPDR0= 172220
682	172222	SDPDR1= 172222
683	172224	SDPDR2= 172224
684	172226	SDPDR3= 172226
685	172230	SDPDR4= 172230
686	172232	SDPDR5= 172232
687	172234	SDPDR6= 172234
688	172236	SDPDR7= 172236
689		
690		;*SUPERVISOR "I" PAGE ADDRESS REGISTERS
691		
692	172240	SIPAR0= 172240
693	172242	SIPAR1= 172242
694	172244	SIPAR2= 172244
695	172246	SIPAR3= 172246
696	172250	SIPAR4= 172250
697	172252	SIPAR5= 172252
698	172254	SIPAR6= 172254
699	172256	SIPAR7= 172256
700		
701		;*SUPERVISOR "D" PAGE ADDRESS REGISTERS
702		
703	172260	SDPAR0= 172260
704	172262	SDPAR1= 172262
705	172264	SDPAR2= 172264
706	172266	SDPAR3= 172266
707	172270	SDPAR4= 172270
708	172272	SDPAR5= 172272
709	172274	SDPAR6= 172274
710	172276	SDPAR7= 172276
711		
712		;*KERNEL "I" PAGE DESCRIPTOR REGISTERS
713		

714	172300	KIPDR0= 172300
715	172302	KIPDR1= 172302
716	172304	KIPDR2= 172304
717	172306	KIPDR3= 172306
718	172310	KIPDR4= 172310
719	172312	KIPDR5= 172312
720	172314	KIPDR6= 172314
721	172316	KIPDR7= 172316

;\*KERNEL "D" PAGE DESCRIPTOR REGISTERS

725	172320	KDPDR0= 172320
726	172322	KDPDR1= 172322
727	172324	KDPDR2= 172324
728	172326	KDPDR3= 172326
729	172330	KDPDR4= 172330
730	172332	KDPDR5= 172332
731	172334	KDPDR6= 172334
732	172336	KDPDR7= 172336

;\*KERNEL "I" PAGE ADDRESS REGISTERS

736	172340	KIPAR0= 172340
737	172342	KIPAR1= 172342
738	172344	KIPAR2= 172344
739	172346	KIPAR3= 172346
740	172350	KIPAR4= 172350
741	172352	KIPAR5= 172352
742	172354	KIPAR6= 172354
743	172356	KIPAR7= 172356

;\*KERNEL "D" PAGE ADDRESS REGISTERS

747	172360	KDPAR0= 172360
748	172362	KDPAR1= 172362
749	172364	KDPAR2= 172364
750	172366	KDPAR3= 172366
751	172370	KDPAR4= 172370
752	172372	KDPAR5= 172372
753	172374	KDPAR6= 172374
754	172376	KDPAR7= 172376

.SBTTL UNIBUS MAP REGISTER DEFINITIONS

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

766	170200	MAPL00 = 170200
767	170202	MAPH00 = 170202
768	170204	MAPL01 = 170204
769	170206	MAPH01 = 170206

770	170210	MAPL02 =	170210
771	170212	MAPH02 =	170212
772	170214	MAPL03 =	170214
773	170216	MAPH03 =	170216
774	170220	MAPL04 =	170220
775	170222	MAPH04 =	170222
776	170224	MAPL05 =	170224
777	170226	MAPH05 =	170226
778	170230	MAPL06 =	170230
779	170232	MAPH06 =	170232
780	170234	MAPL07 =	170234
781	170236	MAPH07 =	170236
782	170240	MAPL10 =	170240
783	170242	MAPH10 =	170242
784	170244	MAPL11 =	170244
785	170246	MAPH11 =	170246
786	170250	MAPL12 =	170250
787	170252	MAPH12 =	170252
788	170254	MAPL13 =	170254
789	170256	MAPH13 =	170256
790	170260	MAPL14 =	170260
791	170262	MAPH14 =	170262
792	170264	MAPL15 =	170264
793	170266	MAPH15 =	170266
794	170270	MAPL16 =	170270
795	170272	MAPH16 =	170272
796	170274	MAPL17 =	170274
797	170276	MAPH17 =	170276
798	170300	MAPL20 =	170300
799	170302	MAPH20 =	170302
800	170304	MAPL21 =	170304
801	170306	MAPH21 =	170306
802	170310	MAPL22 =	170310
803	170312	MAPH22 =	170312
804	170314	MAPL23 =	170314
805	170316	MAPH23 =	170316
806	170320	MAPL24 =	170320
807	170320	MAPH24 =	170320
808	170324	MAPL25 =	170324
809	170326	MAPH25 =	170326
810	170330	MAPL26 =	170330
811	170332	MAPH26 =	170332
812	170334	MAPL27 =	170334
813	170336	MAPH27 =	170336
814	170340	MAPL30 =	170340
815	170342	MAPH30 =	170342
816	170344	MAPL31 =	170344
817	170346	MAPH31 =	170346
818	170350	MAPL32 =	170350
819	170352	MAPH32 =	170352
820	170354	MAPL33 =	170354
821	170356	MAPH33 =	170356
822	170360	MAPL34 =	170360
823	170362	MAPH34 =	170362
824	170364	MAPL35 =	170364
825	170366	MAPH35 =	170366

1

826	170370	MAPL36 = 170370
827	170372	MAPH36 = 170372
828	170374	MAPL37 = 170374
829	170376	MAPH37 = 170376
830		.EQUIV MAPL00,MAPL0
831		.EQUIV MAPH00,MAPH0
832		.EQUIV MAPL01,MAPL1
833		.EQUIV MAPH01,MAPH1
834		.EQUIV MAPL02,MAPL2
835		.EQUIV MAPH02,MAPH2
836		.EQUIV MAPL03,MAPL3
837		.EQUIV MAPH03,MAPH3
838		.EQUIV MAPL04,MAPL4
839		.EQUIV MAPH04,MAPH4
840		.EQUIV MAPL05,MAPL5
841		.EQUIV MAPH05,MAPH5
842		.EQUIV MAPL06,MAPL6
843		.EQUIV MAPH06,MAPH6
844		.EQUIV MAPL07,MAPL7
845		.EQUIV MAPH07,MAPH7

.SBTTL TRAP CATCHER

851		. = 0
852		;
853	000000	;
854		;
855		;
856		;
857		;

.SBTTL STARTING ADDRESS(ES)

858		. = 200
859	000200	;
860		;
861	000200 000137 010000	JMP J*START ; JUMP TO STARTING ADDRESS OF PROGRAM
862		;
863		;

.SBTTL ACT11 HOOKS

864		;
865		;
866		;
867		;
868		;
869		;
870		;
871		;
872		;
873		;
874		;
875		;
876		;
877		;
878		;
879		;
880		;
881		;

882 000204  
883 000046  
884 000046 023100  
885 000052  
886 000052 000000  
887 000204

\$\$VPC=  
.=46  
.WORD SENDAD  
.=52  
.WORD 0  
.\$\$VPC

::: SAVE LOCATION COUNTER  
::: SET LOCATION COUNTER  
::: SET LOC.46 TC ADDRESS SENDAD  
::: SET LOCATION COUNTER  
::: SET LOC.52 TO ZERO  
::: RESTORE LOCATION COUNTER



```

888      ;*****
889
890      .SBTTL  COMMON TAGS
891
892      ;*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
893      ;*USED IN THE PROGRAM.
894
895      001100      .=1100
896
897      001100      $CMTAG:      ;; START OF COMMON TAGS
898      001100      000000      $PASS:      .WORD      0      ;; CONTAINS PASS COUNT
899      001102      000      $STNM:      .BYTE      0      ;; CONTAINS THE TEST NUMBER
900      001103      000      $ERFLG:      .BYTE      0      ;; CONTAINS ERROR FLAG
901      001104      000000      $ICNT:      .WORD      0      ;; CONTAINS SUBTEST ITERATION COUNT
902      001105      000000      $LPADR:      .WORD      0      ;; CONTAINS SCOPE LOOP
903      001110      000000      $LPERR:      .WORD      0      ;; CONTAINS SCOPE RETURN FOR ERRORS
904      001112      000000      $ERTTL:      .WORD      0      ;; CONTAINS TOTAL ERRORS DETECTED
905      001114      000      $ITEMB:      .BYTE      0      ;; CONTAINS ITEM CONTROL BYTE
906      001115      001      $ERMAX:      .BYTE      1      ;; CONTAINS MAX. ERRORS PER TEST
907      001116      000000      $ERRPC:      .WORD      0      ;; CONTAINS PC OF LAST ERROR INSTRUCTION
908      001120      000000      $GDADR:      .WORD      0      ;; CONTAINS OF 'GOOD' DATA
909      001122      000000      $BDADR:      .WORD      0      ;; CONTAINS OF 'BAD' DATA
910      001124      000000      $GDDAT:      .WORD      0      ;; CONTAINS 'GOOD' DATA
911      001126      000000      $BDDAT:      .WORD      0      ;; CONTAINS 'BAD' DATA
912      001130      000000      000000      000000      .WORD      0,0,0      ;; RESERVED--NOT TO BE USED
913      001136      177560      $TKS:      177560      ;; TTY KBD STATUS
914      001140      177562      $TKB:      177562      ;; TTY KBD BUFFER
915      001142      177564      $TPS:      177564      ;; TTY PRINTER STATUS REG.
916      001144      177566      $TPB:      177566      ;; TTY PRINTER BUFFER REG.
917      001146      000      $NULL:      .BYTE      0      ;; CONTAINS NULL CHARACTER FOR FILLS
918      001147      002      $FILLS:      .BYTE      2      ;; CONTAINS # OF FILLER CHARACTERS REQUIRED
919      001150      012      $FILLC:      .BYTE      12      ;; INSERT FILL CHARS. AFTER A "LINE FEED"
920      001151      000      $TPFLG:      .BYTE      0      ;; "TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
921      001152      000000      $REGAD:      .WORD      0      ;; CONTAINS THE FROM
922      ;; WHICH ($REGO) WAS OBTAINED
923      001154      000000      $REGO:      .WORD      0      ;; CONTAINS (($REGAD)+0)
924      001156      000000      $REG1:      .WORD      0      ;; CONTAINS (($REGAD)+2)
925      001160      000000      $REG2:      .WORD      0      ;; CONTAINS (($REGAD)+4)
926      001162      000000      $REG3:      .WORD      0      ;; CONTAINS (($REGAD)+6)
927      001164      000000      $REG4:      .WORD      0      ;; CONTAINS (($REGAD)+10)
928      001166      000000      $REG5:      .WORD      0      ;; CONTAINS (($REGAD)+12)
929      001170      000000      $TMP0:      .WORD      0      ;; USER DEFINED
930      001172      000000      $TMP1:      .WORD      0      ;; USER DEFINED
931      001174      000000      $TMP2:      .WORD      0      ;; USER DEFINED
932      001176      000000      $TMP3:      .WORD      0      ;; USER DEFINED
933      001200      000000      $TMP4:      .WORD      0      ;; USER DEFINED
934      001202      000000      $TMP5:      .WORD      0      ;; USER DEFINED
935      001204      000000      $TIMES:      0      ;; MAX. NUMBER OF ITERATIONS
936      001206      000000      $ESCAPE:      0      ;; ESCAPE ON ERROR
937      001210      177607      000377      $BELL:      .ASCIZ      <207><377><377>      ;; CODE FOR BELL
938      001214      077      $QUES:      .ASCII      /?/      ;; QUESTION MARK
939      001215      015      $CRLF:      .ASCII      <15>      ;; CARRIAGE RETURN
940      001216      000012      $LF:      .ASCIZ      <12>      ;; LINE FEED
941      001220      000000      $PADRSL:      .WORD      0      ;; HOLDS THE LOWER 16 BITS OF A 22 BIT
942      ;; ADDRESS GENERATED FOR TYPE OUT.
943      001222      000000      $PADRSH:      .WORD      0      ;; HOLDS THE UPPER 6 BITS OF A 22 BIT

```

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

986  
987  
988  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
1000  
1001  
1002  
1003  
1004  
1005  
1006  
1007  
1008  
1009  
1010  
1011  
1012  
1013  
1014  
1015  
1016  
1017  
1018  
1019  
1020  
1021  
1022  
1023  
1024  
1025  
1026  
1027  
1028  
1029  
1030  
1031  
1032  
1033  
1034  
1035  
1036  
1037  
1038  
1039  
1040  
1041

```

:*****
.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:
:ITEM 1
EM1      :NOT THE CORRECT CPU TRAP CONDITION THRU ERRVEC (#004)
DH1      :RECEIVD EXPECTD TESTNO PC AT ABORT
DT1      :PCPJR,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
          :PARITY ADDRESS MAINTEN CONTROL
          :CONDITN REFERENC0 REGISTR REGISTR TESTNO PC AT ABORT
DT3      :PPARER,LOADDR,PMaint,PCONTR,TESTNO,BADPC,0
DF3      : 0, 2, 0, 0, 0, 0

:ITEM 4
EM4      :UNEXPECTED MAIN MEMORY PARITY ERROR THRU CACHVEC (#114)
DH4      :WILL RETRY TEST ONCE
          :PARITY ADDRESS MAINTEN CONTROL
          :CONDITN REFERENC0 REGISTR REGISTR TESTNO PC AT ABORT
DT3      :PPARER,LOADDR,PMaint,PCONTR,TESTNO,BADPC,0
DF3      : 0, 2, 0, 0, 0, 0

:ITEM 5
EM5      :MEMORY MANAGEMENT TRAP, MEMORY MANAGEMENT STATUS REGISTERS
DH5      :STATUS AUTOI/D VIRTADR
          :REGISTR REGISTR REGISTR TESTNO PC AT ABORT
DT5      :PMMR0,PMMR1,PMMR2,TESTNO,BADPC,0
DF5      : 0, 0, 0, 0, 0

:ITEM 6
EM6      :SUMMARY OF MAP REGISTERS THAT TIMED OUT ON READ
DH6      :REGADR0 REGADR0
          : "OR" "AND" #ERRORS TESTNO ERRORPC

```

```

00133C
001330 023162
001332 031425
001334 034360
001336 035134

001340 023244
001342 031471
001344 034372
001346 035140

001350 023313
001352 031525
001354 034402
001356 035143

001360 023414
001362 031525
001364 034402
001366 035143

001370 023523
001372 031655

001374 034432
001376 035151

001400 023631
001402 031761

```

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

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

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

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



1266	002002	033312	DH41	:CORRECT ADDRESS
1267				:ADDRESS FETCHED TESTNO ERRORPC
1268	002004	034740	DT41	:\$REG2,\$REG1,TESTNO,\$ERRPC
1269	002006	035263	DF41	: 0, 0, 0, 0
1270				
1271			:ITEM 47	
1272	002010	030264	EM47	:RELOCATION THRU THE MAP WAS NOT CORRECT, CARRY PROPAGATION
1273				:TEST BEING RUN OVER UNIBUS
1274	002012	033372	DH42	:CORRECT EXPECTD RECEIVD
1275				:ADDRESS DATA FROM UB TESTNO ERRORPC
1276	002014	034752	DT42	:\$REG1,\$REG3,\$REG2,TESTNO,\$ERRPC
1277	002016	035267	DF42	: 3, 0, 0, 0, 0
1278				
1279			:ITEM 50	
1280	002020	030417	EM50	:THE TOP OF MEMORY IS DIFFERENT THAN THE SIZE JUMPERS
1281				:TEST BEING RUN OVER UNIBUS
1282	002022	033472	DH43	:SIZJUMP TOPFOUND TESTNO
1283	002024	034766	DT43	:SIZELO,RSIZE,TESTNO,0
1284	002026	035274	DF43	: 0, 0, 0
1285				
1286				
1287			:ITEM 51	
1288	002030	030544	EM51	:PARITY REPORTING THRU THE MAP IS NOT CORRECT
1289				:TEST BEING RUN OVER UNIBUS
1290	002032	033532	DH44	:CONDITN CONDITN ADDRESS MAINTEN CONTROL
1291				:EXPECTD RECEIVD REFERENC'D REGISTR REGISTR TESTNO ERRORPC
1292	002034	035000	DT44	:\$TMP4,\$PPARER,\$PLOADR,\$PMAINT,\$PCONTR,TESTNO,\$ERRPC,0
1293	002036	035300	DF44	: 0, 0, 2, 0, 0, 0, 0
1294				
1295			:ITEM 52	
1296	002040	030661	EM52	:SUMMARY OF DUAL MAPPING ERRORS
1297	002042	032605	DH23	:EXPECTD EXPECTD RECEIVD RECEIVD
1298				: "OR" "AND" "OR" "AND" #ERRORS TESTNO
1299	002044	034710	DT37	:ADDROR,ADRAND,DATAOR,DATAND,\$ERRPC,TESTNO,0
1300	002046	035251	DF37	: 0, 0, 0, 0, 1, 0
1301				
1302	002050		ER200:	:THIS IS THE STARTING POINT FOR ERROR MESSAGES
1303				:201 THRU 377. THEY ARE USED FOR MULTIPLE
1304				:ERROR MESSAGES.
1305				
1306			:ITEM 201	
1307	002050	030720	EM201	:THE FOLLOWING REGISTERS TIMED OUT WHEN READ
1308	002052	033676	DH201	:REGADRS TESTNO ERRORPC
1309	002054	035020	DT201	:\$REG0,TESTNO,\$ERRPC,0
1310	002056	035307	DF201	: 0, 0, 0
1311				
1312			:ITEM 202	
1313	002060	030774	EM202	:THE FOLLOWING MAP REGISTERS WILL NOT CLEAR
1314	002062	033726	DH202	:REGADRS DATAREC TESTNO ERRORPC.
1315	002064	035030	DT202	:\$REG0,\$TMP0,TESTNO,\$ERRPC,0
1316	002066	035312	DF202	: 0, 0, 0, 0
1317				
1318			:ITEM 203	
1319	002070	031047	EM203	:THE FOLLOWING ARE DUAL ADDRESSING ERRORS IN THE UNIBUS MAP
1320	002072	033766	DH203	:MAPREG MAPREG
1321				:TESTING DUALED TESTNO ERRORPC

# H03

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

```

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

```

```

1434      ;*AND GO TO ERTYPE ON ERROR
1435      ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
1436      ;*SW15=1      HALT ON ERROR
1437      ;*          HALT CAN OCCUR BEFORE AND AFTER THE ERROR TYPEOUT
1438      ;*SW13=1      INHIBIT ERROR TYPEOUTS
1439      ;*SW10=1      BELL ON ERROR
1440      ;*SW09=1      LOOP ON ERROR
1441      ;*CALL
1442      ;*      ERROR      N      ;;ERROR=EMT AND N=ERROR ITEM NUMBER
1443
1444      002460      ;ERROR:
1445      002460      113737      001102      001262      MOVB      $TSTNM,TESTNO      ;SAVE TEST NUMBER FOR ERROR TYPE OUT
1446      002466      005237      001254      INC      ERRCNT      ;COUNT ALL MULTIPLE ERRORS
1447      002472      010037      001154      MOV      R0,$REG0      ;SAVE R0 FOR POSSIBLE TYPE OUT
1448      002476      010137      001156      MOV      R1,$REG1      ;SAVE R1 FOR POSSIBLE TYPE OUT
1449      002502      010237      001160      MOV      R2,$REG2      ;SAVE R2 FOR POSSIBLE TYPE OUT
1450      002506      010337      001162      MOV      R3,$REG3      ;SAVE R3 FOR POSSIBLE TYPE OUT
1451      002512      010437      001164      MOV      R4,$REG4      ;SAVE R4 FOR POSSIBLE TYPE OUT
1452      002516      010537      001166      MOV      R5,$REG5      ;SAVE R5 FOR POSSIBLE TYPE OUT
1453      002522      105237      001103      7$:      INCB      $ERFLG      ;SET THE ERROR FLAG
1454      002526      001775      BEQ      7$      ;DON'T LET THE FLAG GO TO ZERO
1455      002530      013737      001102      177570      MOV      $TSTNM,$#DISPLAY ;DISPLAY TEST NUMBER AND ERROR FLAG
1456      002536      005737      177570      TST      $#SWR      ;HALT ON ERROR = 1?
1457      002542      1000C1      BPL      8$      ;BRANCH IF NO
1458      002544      000000      HALT      ;YES--HALT
1459      002546      032737      002000      177570      8$:      BIT      #BIT10,$#SWR      ;BELL ON ERROR?
1460      002554      001402      BEQ      1$      ;NO - SKIP
1461      002556      104400      001210      TYPE      $BELL      ;RING BELL
1462      002562      005237      001112      1$:      INC      $ERTTL      ;COUNT THE NUMBER OF ERRORS
1463      002566      011637      001116      MOV      (SP),$ERRPC      ;GET ADDRESS OF ERROR INSTRUCTION
1464      002572      162737      000002      001116      SUB      #2,$ERRPC
1465      002600      117737      176312      001114      MOV      $#ERRPC,$ITEMB ;STRIP AND SAVE THE ERROR ITEM CODE
1466      002606      032737      020000      177570      BIT      #BIT13,$#SWR      ;SKIP TYPEOUT IF SET
1467      002614      001004      BNE      2$      ;SKIP TYPEOUTS
1468      002616      004737      002762      JSR      PC,ERTYPE      ;GO TO USER ERROR ROUTINE
1469      002622      104400      001215      TYPE      $,RLF
1470      002626      005737      177570      2$:      TST      $#SWR      ;HALT ON ERROR
1471      002632      100001      BPL      9$      ;SKIP IF CONTINUE
1472      002634      000000      HALT      ;HALT ON ERROR!
1473      002636      022737      023100      000042      9$:      CMP      #SENDAD,42      ;ACT-11?
1474      002644      001001      BNE      3$      ;BRANCH IF NO
1475      002646      000000      HALT      ;YES
1476      002650      032737      001000      177570      3$:      BIT      #BIT09,$#SWR      ;LOOP ON ERROR SWITCH SET?
1477      002656      001402      BEQ      4$      ;BR IF NO
1478      002660      013716      001110      MOV      $LPERR,(SP)      ;FUDGE RETURN FOR LOOPING
1479      002664      005737      001206      4$:      TST      $ESCAPE      ;CHECK FOR AN ESCAPE ADDRESS
1480      002670      001402      BEQ      5$      ;BR IF NONE
1481      002672      013716      001206      MOV      $ESCAPE,(SP)      ;FUDGE RETURN ADDRESS FOR ESCAPE
1482      002676      032737      001000      177570      5$:      BIT      #SW9,SWR      ;ARE YOU LOOPING ON THIS ERROR?
1483      002704      001425      BEQ      EEXIT      ;BRANCH IF NOT LOOPING ON ERROR
1484      002706      012737      177777      177766      MOV      #-1,$#CPUERR      ;CLEAR CPU ERROR REGISTER
1485      002714      012737      177777      177744      MOV      #-1,$#MEMERR      ;CLEAR MEMORY ERROR REGISTER
1486      002722      042737      177776      177572      BIC      #177776,$#MMRO      ;CLEAR MEMORY MANAGEMENT STATUS REGISTER
1487      002730      012737      177777      005074      MOV      #-1,$TOFLAG      ;INITIALIZE TRAP FLAG
1488      002736      012737      177777      005574      MOV      #-1,$CPFLAG      ;INITIALIZE CP TRAP FLAG

```

```

1490 002744 012737 177777 005706      MOV    #-1,PAFLAG      ;INITIALIZE PARITY TRAP FLAG
1491 002752 012737 177777 006132      MOV    #-1,MMFLAG     ;INITIALIZE MEMORY MANAGEMENT TRAP FLAG
1492 002760 000002                EREXIT: RTI           ;RETURN TO TEST
    
```

.SBTTL ERROR MESSAGE TYPE OUT ROUTINE

```

1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519 002762 010046
1520 002764 005000
1521 002766 113700 001114
1522 002772 001004
1523 002774 013746 001116
1524 003000 104402
1525 003002 000526
1526 003004 005300
1527 003006 072027 000003
1528 003012 100024
1529 003014 032700 001000
1530 003020 001415
1531
1532 003022 032737 000200 177570
1533 003030 001404
1534
1535 003032 062766 000004 000002
1536
1537 003040 000507
1538 003042 042700 177000
1539 003046 062700 002054
1540 003052 000424
1541 003054 042700 177000
1542 003060 062700 000520
1543
1544
1545

; * THIS SUBROUTINE IS CALLED BY THE ERROR HANDLER TO TYPE
; * THE ERROR MESSAGES. IT PICKS UP THE ITEM BYTE ($ITEMB) NUMBER
; * AND USES THAT TO INDEX THROUGH THE ERROR TABLE. THE ERROR
; * TABLE STARTS AT "$ERRTB" AND HAS FOUR (4) POINTERS FOR EACH
; * ENTRY. 'EM' 'DH' 'DT' 'DF'. THE 'EM' POINTS TO THE ERROR
; * MESSAGE WHICH IS AN ASCIZ STRING. THE 'DH' POINTS TO THE DATA
; * HEADER WHICH IS ANOTHER ASCIZ STRING. THE 'DT' POINTS TO THE
; * DATA TABLE WHICH IS A GROUP OF WORDS CONTAINING THE ADDRESSES
; * OF THE DATA TO BY TYPED. THE FORMAT OF THIS DATA IS
; * CONTROLLED BY THE 'DF' WHICH IS THE POINTER TO THE DATA FORMAT.
; * THE DATA FORMAT IS A GROUP OF BYTES WHICH CONTAIN NUMBERS
; * THAT CORRESPOND TO DIFFERENT TYPING FORMATS.
; *
; * 0 -16 BIT OCTAL FORMAT
; * 1 -DECIMAL FORMAT
; * 2 -22 BIT OCTAL FORMAT. DATA IS LOWER 16 BITS OF THE
; * PHYSICAL ADDRESS, UPPER 6 BITS ARE ADJACENT TO LOWER 16
; * 3 -22 BIT OCTAL FORMAT. DATA IS THE 16 BIT VIRTUAL
; * ADDRESS IN KERNEL I-SPACE.
; * 4 -18 BIT OCTAL FORMAT. DATA IS A 16 BIT NUMBER THAT
; * WILL BE CONVERTED INTO A UNIBUS ADDRESS BY LEFT
; * SHIFTING IT 6 BITS.
ERTYPE: MOV    RO,-(KSP)      ;SAVE RO ON STACK
        CLR    RO           ;CLEAR RO
        MOVB   2*$ITEMB,RO  ;PUT ITEM NUMBER IN RO
        BNE   1$           ;BRANCH IF IT IS NON-ZERO
        MOV    $ERRPC,-(KSP) ;PUT ERROR PC ON STACK FOR TYPING
        TYPOC ;TYPE FAILING PC
        BR    13$          ;GO TO RETURN
1$:     DEC    RO           ;ADJUST ITEM NUMBER TO BE A POINTER
        ASH   #3,RO        ;LEFT SHIFT ITEM NO. 3 PLACES
        BPL   22$          ;BRANCH IF ITEM # LESS THAN 200
        BIT   #BIT9,RO     ;SEE IF ITEM # WAS 3XX
        BEQ   21$          ;BRANCH IF ITEM # WAS 2XX
        ;AND TYPE ERROR MESSAGE
        ;SEE IF SWITCH 7 IS UP
        BEQ   20$          ;BRANCH IF SWITCH IS NOT UP
        ;AND TYPE DATA, ON MULTIPLE ERRORS
        ADD   #4,2(KSP)    ;SKIP 'TYPE $CRLF' IF SW 7 IS UP
        ;INHIBIT MULTIPLE ERROR TYPEOUTS
        BR    13$          ;BRANCH TO EXIT
20$:   BIC   #177000,RO     ;CLEAR UPPER BYTE OF RO
        ADD   #ER200+4,RO  ;POINT TO DATA TABLE ENTRY
        BR    5$           ;GO TYPE DATA TABLE
21$:   BIC   #177000,RO     ;CLEAR UPPER BYTE OF RO
        ADD   #<ER200-$ERRTB>,RO ;ADD DIFFERENCE BETWEEN
        ;ITEM 1 AND ITEM 201
        ; GET POINTER TO ERROR MESSAGE AND TYPE IT
        ; IF THE POINTER IS NOT ZERO
    
```



# M03

1602	003244	104400	003264	TYPE	32\$	;TYPE TWO SPACES
1603	003250	005711		TST	(R1)	;IS THERE ANOTHER WORD?
1604	003252	001401		BEQ	12\$	;BRANCH IF ALL DONE
1605	003254	000727		BR	6\$	;GO BACK FOR NEXT NUMBER
1606	003256	012601		12\$: MOV	(KSP)+,R1	;RESTORE R1
1607	003260	012600		13\$: MOV	(KSP)+,R0	;RESTORE R0
1608	003262	000207		RTS	PC	;RETURN TO ERROR ROUTINE
1609	003264	020040	000	32\$: .ASCIZ	? ?	;TWO SPACES
1610		003270		.EVEN		
1611						
1612						
1613				.SBTTL	CONVERT 16-BIT VIRTUAL ADDRESS TO 22-BIT PHYSICAL ADDRESS	
1614				;*		
1615				;*	THIS ROUTINE IS CALLED BY A 'JSR PC' AFTER THE VIRTUAL ADDRESS	
1616				;*	IS PUSHED ON THE KERNEL STACK. THE V.A. IS THEN LOADED INTO	
1617				;*	R1 AND THE UPPER 3 BITS ARE SHIFTED INTO R0 TO SELECT THE	
1618				;*	CORRECT KERNEL I-SPACE PAR. THE LOWER 12 BITS OF THE VIRTUAL	
1619				;*	ADDRESS ARE ADDED TO THE PAR AS THEY ARE BY MEMORY MANAGEMENT	
1620				;*	AND THE PHYSICAL ADDRESS IS SAVED IN MEMORY TO BE CONVERTED	
1621				;*	TO ASCIZ AND TYPED.	
1622				;*		
1623	003270	104412		TYPVAD: SAVREG		;SAVE ALL REGISTERS
1624	003272	016601	000002	MOV	2(KSP),R1	;PUT VIRTUAL ADDR IN R1
1625	003276	005000		CLR	R0	;CLEAR R0 FOR CALCULATIONS
1626	003300	073027	000003	ASHC	#3,R0	;LEFT SHIFT R0,R1 3 PLACES
1627	003304	006300		ASL	R0	;LEFT SHIFT R0 ONE MORE PLACE
1628	003306	006001		ROR	R1	;RIGHT SHIFT R1 SO OFFSET IS CORRECT
1629	003310	006001		ROR	R1	;RIGHT SHIFT R1
1630	003312	006001		ROR	R1	;RIGHT SHIFT R1
1631	003314	062700	172340	ADD	#KIPAR0,R0	;FORM DESIRED PAR ADDR IN R0
1632	003320	011003		MOV	(R0),R3	;PUT CONTENTS OF PAR IN R3
1633	003322	005002		CLR	R2	;CLEAR R2 FOR PHYSICAL ADDR CALCULATIONS
1634	003324	073227	000006	ASHC	#6,R2	;LEFT SHIFT (R2,R3) 6 PLACES
1635	003330	060103		ADD	R1,R3	;ADD OFFSET IN R1 TO BASE IN R3
1636	003332	005502		ADC	R2	;ADD ANY POSSIBLE CARRY TO UPPER 6 BITS
1637	003334	010237	001222	MOV	R2,PADRSR	;PUT UPPER 6 BITS OF ADDR IN CORE
1638	003340	010337	001220	MOV	R3,PADRSL	;PUT LOWER 16 BITS OF ADDR IN CORE
1639	003344	012746	001220	MOV	#PADRSL,-(KSP)	;PUT POINTER TO LOWER 16 BITS ON STACK
1640	003350	004737	004670	JSR	PC,\$DB20	;CONVERT NUMBER TO OCTAL ASCIZ
1641	003354	062716	000003	ADD	#3,(KSP)	;ONLY TYPE 8 DIGITS
1642	003360	012637	003366	MOV	(KSP)+,3\$	;PUT POINTER AFTER TYPE INST
1643	003364	104400		TYPE		;TYPE THE 22-BIT VIRTUAL ADDRESS
1644	003366	000000		3\$: .WORD	0	;THIS WORD HOLDS THE POINTER TO
1645						THE ASCIZ STRING
1646	003370	104414		RESREG		;RESTORE ALL THE REGISTERS
1647	003372	012616		MOV	(KSP)+,(KSP)	;LEAVE ONLY RETURN ADDR ON STACK
1648	003374	000207		RTS	PC	;RETURN TO ERROR HANDLER
1649						
1650						
1651				;*	THIS SUBROUTINE IS USED TO CONVERT THE A WORD PUSHED	
1652				;*	ON THE STACK INTO A UNIBUS ADDRESS AND TYPE IT AS A	
1653				;*	#6 DIGIT NUMBER. IT USES R1 & R0 AND LEAVES	
1654				;*	ALL OTHER REGISTERS UNCHANGED.	
1655	003376	016601	000002	UBADDR: MOV	2(KSP),R1	;LOAD 16 BIT ADDRESS INTO R1
1656	003402	005000		CLR	R0	;CLEAR R0 FOR CALCULATIONS
1657	003404	073027	000006	ASHC	#6,R0	;LEFT SHIFT (R0,R1) 6 PLACES



# NO3

PDP-11/70 UNIBUS MAP DIAGNOSTIC MACY11 27(732) 20-SEP-76 13:32 PAGE 39  
 DEKBFB.P11 CONVERT 16-BIT VIRTUAL ADDRESS TO 22-BIT PHYSICAL ADDRESS

```

1658 003410 010137 001220      MOV      R1,PADRSL      ;PUT LOWER 16 BITS IN PADRS
1659 003414 010037 001222      MOV      R0,PADRSH      ;PUT UPPER 6 BITS IN PADRS
1660 003420 012746 001220      MOV      #PADRSL, -(KSP) ;PUSH POINTER TO WORDS ON STACK
1661 003424 004737 004670      JSR      PC,$DB20      ;JUMP TO CONVERT ROUTINE
1662 003430 062716 000005      ADD      #5,(KSP)      ;ONLY USE LOWER 6 CHARS.
1663 003434 012637 003442      MOV      (KSP)+,3$     ;PUT POINTER AFTER TYPE CALL.
1664 003440 104400
1665 003442 000000      3$:      .WORD      0      ;HOLDS POINTER TO FIRST CHAR.
1666 003444 012616      MOV      (KSP)+,(KSP)  ;LEAVE ONLY RETURN ADDRESS ON STACK
1667 003446 000207      RTS      PC            ;RETURN TO ERROR TYPE ROUTINE.
  
```

;;\*\*\*\*\*

.SBTTL SAVE AND RESTORE R0-R5 ROUTINES

```

; *SAVE R0-R5
; *CALL:
; *      SAVREG
; *UPON RETURN FROM $SAVREG THE STACK WILL LOOK LIKE:
; *
; *TOP---(+16)
; * +2---(+18)
; * +4---R5
; * +6---R4
; * +8---R3
; *+10---R2
; *+12---R1
; *+14---R0
  
```

\$SAVREG:

```

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
  
```

```

1688 003450
1689 003450 010046
1690 003452 010146
1691 003454 010246
1692 003456 010346
1693 003460 010446
1694 003462 010546
1695 003464      646 000022
1696 003470      0646 000022
1697 003474 J16646 000022
1698 003500 016646 000022
1699 003504 000002
1700
1701
1702
1703
1704 003506
1705 003506 012666 000022
1706 003512 012666 000022
1707 003516 012666 000022
1708 003522 012666 000022
1709 003526 012605
1710 003530 012604
1711 003532 012603
1712 003534 012602
1713 003536 012601
  
```

```

1714 003540 012600          MOV      (SP)+,RO      ;;POP STACK INTO RO
1715 003542 000002          RTI
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
003544 105737 00115:          STYPE:  TSTB      $TPFLG      ;; IS THERE A TERMINAL?
003550 100002          BPL      IS                ;; BR IF YES
003552 000000          HALT
003554 000407          BR      3$                ;; HALT HERE IF NO TERMINAL
003556 010046          1$:      MOV      RO,-(SP)    ;; LEAVE
003560 017600 000002          MOV      22(SP),RO        ;; SAVE RC
003564 112046          2$:      MOVB     ,(RO)+,-(SP)  ;; GET ADDRESS OF ASCIZ STRING
003566 001005          BNE     4$                ;; PUSH CHARACTER TO BE TYPED ONTO STACK
003570 005726          TST     (SP)+             ;; BR IF IT ISN'T THE TERMINATOR
003572 012600          MOV     (SP)+,RO          ;; IF TERMINATOR POP IT OFF THE STACK
003574 062716 000002          3$:      ADD      #2,(SP)        ;; RESTORE RO
003600 000002          RTI                        ;; ADJUST RETURN PC
003602 122716 000011          4$:      CMPB     #HT,(SP)       ;; RETURN
003606 001424          BEQ     8$                ;; BRANCH IF 'HT'
003610 122716 000200          CMPB     #CRLF,(SP)       ;; BRANCH IF NOT
003614 001004          BNE     5$                ;; POP (CR)(LF) EQUIV
003616 005726          TST     (SP)+             ;; GET NEXT CHARACTER
003620 104400 001215          TYPE     $CRLF           ;; GO TYPE THIS CHARACTER
003624 000757          BR      2$                ;; IS IT TIME FOR FILLER CHARS.?
003626 004737 003704          5$:      JSR      PC,$TYPEC      ;; IF NO GO GET NEXT CHAR.
003632 123726 001150          6$:      CMPB     $FILLC,(SP)+    ;; GET # OF FILLER CHARS. NEEDED
003636 001352          BNE     2$                ;; AND THE NULL CHAR.
003640 013746 001146          MOV     $NULL,-(SP)       ;; DOES A NULL NEED TO BE TYPED?
003644 105366 000001          7$:      DECB     1(SP)          ;; BR IF NO--GO POP THE NULL OFF OF STACK
003650 002770          BLT     6$                ;; GO TYPE A NULL
003652 004737 003704          JSR      PC,$TYPEC
003656 000772          BR      7$                ;; LOOP
1767
1768
1769
003660 112716 000040          ;;HORIZONTAL TAB PROCESSOR
8$:      MOVB     #' ,(SP)    ;;REPLACE TAB WITH SPACE

```

```

1770 003664 004737 003704 95: JSR PC,$TYPEC ;;TYPE A SPACE
1771 003670 132737 000007 003750 BITB #7,$CHARCNT ;;BRANCH IF NOT AT
1772 003676 001372 BNE 95 ;;TAB STOP
1773 003700 005726 TST (SP)+ ;;POP SPACE OFF STACK
1774 003702 000730 BR 25 ;;GET NEXT CHARACTER
1775 003704 105777 175232 $TYPEC: TSTB $STPS ;;WAIT UNTIL PRINTER IS READY
1776 003710 100375 SPL $TYPEC
1777 003712 116677 000002 175224 MOVB 2(SP),$STPB ;;LOAD CHAR TO BE TYPED INTO DATA REG.
1778 003720 122766 000015 000002 CMPB #CR,2(SP) ;;BRANCH IF
1779 003726 001003 BNE 15 ;;NOT <CR>
1780 003730 105037 003750 CLRB $CHARCNT
1781 003734 000406 BR $TYPEX ;;EXIT
1782 003736 122766 000012 000002 15: CMPB #LF,2(SP) ;;BRANCH IF
1783 003744 001402 BEQ $TYPEX ;;<LF>
1784 003746 105227 INCB (PC)+ ;;INC SPACE
1785 003750 000000 $CHARCNT: .WORD 0 ;;COUNT
1786 003752 000207 $TYPEX: RTS PC
1787
1788 ;;*****
1789
1790 .SBTTL BINARY TO OCTAL (ASCII) AND TYPE
1791
1792 *THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
1793 *OCTAL (ASCII) NUMBER AND TYPE IT.
1794 *$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
1795 *CALL:
1796 * MOV NUM,-(SP) ;;NUMBER TO BE TYPED
1797 * TYPOS ;;CALL FOR TYPEOUT
1798 * .BYTE N ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
1799 * .BYTE M ;;M=1 OR 0
1800 * ;;1=TYPE LEADING ZEROS
1801 * ;;0=SUPPRESS LEADING ZEROS
1802
1803 *$STYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
1804 *$TYPOS OR $TYPOC
1805 *CALL:
1806 * MOV NUM,-(SP) ;;NUMBER TO BE TYPED
1807 * TYPON ;;CALL FOR TYPEOUT
1808
1809 *$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
1810 *CALL:
1811 * MOV NUM,-(SP) ;;NUMBER TO BE TYPED
1812 * TYPOC ;;CALL FOR TYPEOUT
1813
1814 003754 017646 000000 $TYPOS: MOV 2(SP),-(SP) ;;PICKUP THE MODE
1815 003760 116637 000001 004177 MOVB 1(SP),$OFILL ;;LOAD ZERO FILL SWITCH
1816 003766 112637 004201 MOVB (SP)+,$OMODE+1 ;;NUMBER OF DIGITS TO TYPE
1817 003772 062716 000002 ADD #2,(SP) ;;ADJUST RETURN ADDRESS
1818 003776 000406 BR $TYPON
1819 004000 112737 000001 004177 $TYPOC: MOVB #1,$OFILL ;;SET THE ZERO FILL SWITCH
1820 004006 112737 000006 004201 MOVB #6,$OMODE+1 ;;SET FOR SIX(6) DIGITS
1821 004014 112737 000005 004176 $TYPON: MOVB #5,$OCNT ;;SET THE ITERATION COUNT
1822 004022 010346 MOV R3,-(SP) ;;SAVE R3
1823 004024 010446 MOV R4,-(SP) ;;SAVE R4
1824 004026 010546 MOV R5,-(SP) ;;SAVE R5
1825 004030 113704 004201 MOVB $OMODE+1,R4 ;;GET THE NUMBER OF DIGITS TO TYPE

```

```

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

```

```

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

```

1938  
1939  
1940  
1941  
1942  
1943 004426 010046  
1944 004430 016600 000002  
1945 004434 005740  
1946 004436 111000  
1947 004440 016000 004446  
1948 004444 000200  
1949  
1950  
1951  
1952  
1953  
1954  
1955  
1956  
1957  
1958 004446  
1959 004446 003544  
1960 004450 004000  
1961 004452 003754  
1962 004454 004014  
1963 004456 004202  
1964 004460 003450  
1965 004462 003506  
1966 004464 005010  
1967 004466 005036  
1968  
1969  
1970  
1971  
1972  
1973  
1974 004470 012737 004616 000024  
1975 004476 012737 000340 000026  
1976 004504 010046  
1977 004506 010146  
1978 004510 010246  
1979 004512 010346  
1980 004514 010446  
1981 004516 010546  
1982 004520 010637 004622  
1983 004524 012737 004536 000024  
1984 004532 000000  
1985 004534 000776  
1986  
1987  
1988 004536 013706 004622  
1989 004542 005037 004622  
1990 004546 005237 004622  
1991 004552 001375  
1992 004554 012605  
1993 004556 012604

;\*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 RO, -(SP) ;; SAVE RO
        MOV 2(SP), RO ;; GET TRAP ADDRESS
        TST -(RO) ;; BACKUP BY 2
        MOVB (RO), RO ;; GET RIGHT BYTE OF TRAP
        MOV $TRPAD(RO), RO ;; INDEX TO TABLE
        RTS RO ;; GO TO ROUTINE
```

.SBTTL TRAP TABLE

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

ROUTINE  
-----

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

;;\*\*\*\*\*

.SBTTL POWER DOWN AND UP ROUTINES

:POWER DOWN ROUTINE

```
$PWRDN: MOV $SILLUP, $PWRVEC ;; SET FOR FAST UP
        MOV $340, $PWRVEC+2 ;; PRIO:7
        MOV RO, -(SP) ;; PUSH RO 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 $SPWRUP, $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
        IS: INC $$SAVR6 ;; WAIT FOR THE INC
        BNE IS OF WORD
        MOV (SP)+, R5 ;; POP STACK INTO R5
        MOV (SP)+, R4 ;; POP STACK INTO R4
```

```

1994 004560 012603      MOV      (SP)+,R3      ;; POP STACK INTO R3
1995 004562 012602      MOV      (SP)+,R2      ;; POP STACK INTO R2
1996 004564 012601      MOV      (SP)+,R1      ;; POP STACK INTO R1
1997 004566 012600      MOV      (SP)+,R0      ;; POP STACK INTO R0
1998 004570 012737 004470 000024      MOV      *$PWRDN,@*PWRVEC ;; SET UP THE POWER DOWN VECTOR
1999 004576 012737 000340 000026      MOV      #340,@*PWRVEC+2 ;; PRIO:7
2000 004604 104400      TYPE                                ;; REPORT THE POWER FAILURE
2001 004606 004624      $PWRMG: .WORD      PWRMSG      ;; POWER FAIL MESSAGE POINTER
2002 004610 012716      MOV      (PC)+,(SP)      ;; RESTART AT START
2003 004611 010000      $PWRAD: .WORD      START      ;; RESTART ADDRESS
2004 004614 000002      RTI
2005 004616 000000      $ILLUP: HALT          ;; THE POWER UP SEQUENCE WAS STARTED
2006 004620 000776      BR      .-2              ;; BEFORE THE POWER DOWN WAS COMPLETE
2007 004622 000000      $$SAVR6: 0              ;; PUT THE SP HERE
2008 004624 006412 047520 042527      PWRMSG: .ASCIZ      <12><15>'POWER FAILURE, RESTARTING PROGRAM?'
2009 004632 020122 040506 046111
2010 004640 051125 026105 051040
2011 004646 051505 040524 052122
2012 004654 047111 020107 051120
2013 004662 043517 040522 000115

      .EVEN

;; *****
.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,@*$DB20      ;; CALL THE ROUTINE
;*      RETURN      ;; THE ADDRESS OF THE FIRST ASCII CHAR. IS ON THE STACK

$DB20: SAVREG      ;; SAVE ALL REGISTERS
      MOV      2(SP),R1      ;; PICKUP THE POINTER TO LOW WORD
      MOV      #$OCTVL+13.,R5 ;; POINTER TO DATA TABLE
      MOV      #12.,R4      ;; DO ELEVEN CHARACTERS
      MOV      #1C7,R3      ;; MASK
      MOV      (R1)+,R0      ;; LOWER WORD
      MOV      (R1)+,R1      ;; HIGH WORD
      CLR      R2          ;; TERMINATOR
15:   MOVB      R2,-(R5)      ;; PUT CHARACTER IN DATA TABLE
      MOV      R0,R2      ;; GET THIS DIGIT
      DEC      R4          ;; COUNT THIS CHARACTER
      BGT      3$        ;; BR IF NOT THE LAST DIGIT
      BEQ      2$        ;; 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
25:   ASR      R3          ;; POSITION THE MASK FOR THE LAST DIGIT
35:   ROR      R1          ;; POSITION THE BINARY NUMBER FOR
      ROR      R0          ;; THE NEXT OCTAL DIGIT
      ROR      R1
      ROR      R0

```

2050 004756 006001  
2051 004760 006000  
2052 004762 040302  
2053 004764 062702 000060  
2054 004770 000753  
2055 004772 000016

RUR R1  
ROR R0  
BIC R3,R2 ;:MASK OUT ALL JUNK  
ADD #0,R2 ;:MAKE THIS CHAR. ASCII  
BR 15 ;:GO PUT IT IN THE DATA TABLE  
\$OCTVL: .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.  
\*\*\*\*\*

2071  
2072 005010  
2073 005010 032766 000020 000002  
2074 005016 001406  
2075 005020 016637 000002 001310  
2076 005026 042766 000020 000002  
2077 005034 000006

\*\*\*\*\*  
\* .EQUIV BIT4,TBIT ;T-BIT IS BIT04 IN PROC. STATUS  
TBITOFF:  
BIT #TBIT,2(KSP) ;IS THE T-BIT ON?  
BEQ 15 ;BRANCH TO EXIT : 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  
15: 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'.  
\*\*\*\*\*

2091 005036  
2092 005036 013766 001310 000002  
2093 005044 042737 000020 001310  
2094  
2095 005052 000006  
2096  
2097  
2098  
2099

\*\*\*\*\*  
\* 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 MAP000 INTO R3 AND THEN CLEARING THE  
\* REGISTER POINTED TO BY R3 UNTIL R3 POINTS ABOVE MAPH37.  
\*\*\*\*\*

2100  
2101  
2102  
2103  
2104  
2105



```

2106
2107
2108 005054 012703 170200 CLRMAP: MOV #MAPLO,R3 ;PUT FIRST MAP ADDR IN R3
2109 005060 005023 1$: CLR (R3)+ ;CLEAR MAP REGS
2110 005062 022703 170376 CMP #MAPH37,R3 ;SEE IF LAST ADDR IS IN R3
2111 005066 103374 BHIS 1$ ;BRANCH IF R3 LE THAN LAST ADDR
2112 005070 000207 RTS PC ;RETURN TO MAIN PROGRAM

```

```

2113
2114 .SBTTL SUBROUTINE TO LOG AND REPORT TIMEOUTS OF MAP REGISTERS
2115
2116
2117
2118
2119
2120
2121
2122

```

```

2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131

```

```

2123 005072
2124 005072 005227
2125 005074 177777
2126 005076 001401
2127 005100 000000
2128
2129
2130
2131

```

```

2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161

```

```

2132 005102 012637 001304 10$: MOV (KSP)+,OLDPC ;SAVE RETURN ADDRESS
2133 005106 012637 001306 MOV (KSP)+,OLDPS ;SAVE OLD PSW
2134 005112 013737 177766 001266 MOV CPUERR,PCPUER ;SAVE CPU ERROR REGISTER
2135 005120 013737 001266 177766 MOV PCPUER,CPUERR ;CLEAR CPU ERROR REGISTER
2136 005126 023737 001266 001264 CMP PCPUER,CPUEXP ;SEE IF EXPECTED CONDITION CAME UP.
2137 005134 001402 BEQ 1$ ;BRANCH IF IT WAS A TIMEOUT
2138 005136 104001 ERROR 1 ;NOT RIGHT CONDITION
2139 005140 000414 BR 3$ ;BRANCH TO EXIT

```

```

2140 005142 050037 001226 1$: BIS RO,ADDROR ;PERFORM LOGIAL OR OF FAILING ADDRESS
2141 005146 005100 COM RO ;GET RO READY FOR AND
2142 005150 040037 001224 BIC RO,ADRAND ;PERFORM LOGICAL AND
2143 005154 005100 COM RO ;PUT RO BACK AS IT WAS
2144 005156 005737 001254 TST ERRCNT ;IS THIS THE FIRST ERROR
2145 005162 001002 BNE 2$ ;BRANCH IF NOT FIRST ERROR
2146 005164 104201 ERROR 201 ;NO REGISTER RESPONSE.
2147 005166 000401 BR 3$ ;BRANCH TO EXIT

```

```

2148 005170 104301 2$: ERROR 301 ;CONTINUE NO RESPONSE TABLE
2149 005172 012737 177777 005074 3$: MOV #-1,TOFLAG ;RESET ONE TIME GATE
2150 005200 013746 001306 MOV OLDPS,-(KSP) ;RESTORE OLD PSW
2151 005204 013746 001304 MOV OLDPC,-(KSP) ;PUSH RETURN ADDRESS BACK ON THE STACK
2152 005210 000006 RTT ;RETURN TO THE TEST

```

```

2153
2154 .SBTTL SUBROUTINE TO REPORT MAP REGISTERS THAT WILL NOT HOLD ZERO
2155
2156
2157
2158
2159
2160
2161

```

```

2157
2158
2159
2160
2161

```

```

2162
2163
2164 005212
2165 005212 012637 001304
2166 005216 050037 001226
2167 005222 005100
2168 005224 040037 001224
2169 005230 005100
2170 005232 053737 001170 001232
2171 005240 005137 001170
2172 005244 043737 001170 001230
2173 005252 005137 001170
2174 005256 005737 001254
2175 005262 001002
2176 005264 104202
2177 005266 000401
2178 005270 104302
2179 005272 000177 174006
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191 005276
2192 005276 012637 001304
2193 005302 050037 001226
2194 005306 005100
2195 005310 040037 001224
2196 005314 005100
2197 005316 050137 001232
2198 005322 005101
2199 005324 040137 001230
2200 005330 005101
2201 005332 005737 001254
2202 005336 001002
2203 005340 104203
2204 005342 000401
2205 005344 104303
2206 005346 000177 173732
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217

```

```

;*
;*****
WRITEERROR:
MOV      (KSP)+,OLDPC      ;SAVE RETURN ADDRESS IN CASE OF ERROR LOOP
BIS      RO,ADDROR        ;LOGICAL "OR" OF BAD ADDRESS
COM      RO                ;GET RO READY FOR AND
BIC      RO,ADRAND        ;PERFORM LOGICAL AND
COM      RO                ;PUT RO BACK AS IT WAS
BIS      $TMP0,DATAOR     ;LOGICAL "OR" OF BAD DATA
COM      $TMP0            ;GET $TMP0 READY FOR AND
BIC      $TMP0,DATAND     ;PERFORM LOGICAL AND
COM      $TMP0            ;PUT $TMP0 BACK AS IT WAS
TST      ERRCNT           ;SEE IF THIS IS FIRST ERROR
BNE      1$               ;BRANCH IF NOT FIRST ERROR
ERROR    202              ;ERROR TYPE OUT ITEM 4
BR       2$               ;SKIP NEXT STATEMENT
1$:      ERROR            302 ;ERROR TYPE OUT ITEM 5
2$:      JMP              @OLDPC ;RETURN TO THE TEST

```

```

.SBTTL SUBROUTINE TO REPORT DUAL ADDRESSING WHEN LOADING A MAP REGISTER
;*****
;
; THIS SUBROUTINE WILL LOG AND REPORT ALL DUAL ADDRESSING ERRORS
; FOUND IN THE MAPPING REGISTERS. A "LOGICAL OR" AND A
; "LOGICAL AND" OF THE WRITTEN ADDRESSES WILL BE MAINTAINED IN
; 'ADDROR' AND 'ADRAND'. THE LOG ON THE ADDITIONAL OR FAILING,
; ADDRESSES WILL BE MAINTAINED IN 'DATAOR' AND 'DATAND'.
;
;*****

```

```

;*****
DUALADR:
MOV      (KSP)+,OLDPC      ;STARTING LOCATION OF SUBROUTINE
BIS      RO,ADDROR        ;SAVE RETURN ADDRESS IN CASE OF LOOP
COM      RO                ;LOGICAL OR OF WRITTEN ADDRESS
BIC      RO,ADRAND        ;GET RO READY FOR AND
COM      RO                ;PERFORM LOGICAL AND
BIS      R1,DATAOR        ;PUT RO 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

```

```

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

```



COPY

# K04

```

2218 005352 012637 001304      COUNT:  MOV      (KSP)+,OLDPC      ;SAVE RETURN ADDRESS IN CASE OF ERROR LOOP
2219 005356 050037 001226      BIS      R0,ADDROR      ;LOGICAL OR OF BAD ADDRESSES
2220 005362 005100      COM      R0              ;GET R0 READY FOR AND
2221 005364 040037 001224      BIC      R0,ADRAND      ;PERFORM LOGICAL AND
2222 005370 005100      COM      R0              ;PUT R0 BACK AS IT WAS
2223 005372 050337 001232      BIS      R3,DATAOR      ;LOGICAL OR OF BAD DATA
2224 005376 005103      COM      R3              ;GET R3 READY FOR AND
2225 005400 040337 001230      BIC      R3,DATAND      ;PERFORM LOGICAL AND
2226 005404 005103      COM      R3              ;PUT R3 BACK AS IT WAS
2227 005406 050437 001236      BIS      R4,PATTOR      ;LOGICAL OR OF PATTERN LOADED
2228 005412 005104      COM      R4              ;GET R4 READY FOR AND
2229 005414 040437 001234      BIC      R4,PATAND      ;PERFORM LOGICAL AND
2230 005420 005104      COM      R4              ;PUT R4 BACK AS IT WAS
2231 005422 005737 001254      TST      ERRCNT          ;IS THIS FIRST ERROR
2232 005426 001002      BNE      1$             ;BRANCH IF NOT FIRST ERROR
2233 005430 104204      ERROR   204            ;REPORT FIRST COUNT ERROR
2234 005432 000401      BR       2$             ;SKIP NEXT STATEMENT
2235 005434 104304      1$:     ERROR   304            ;REPORT MORE DATA
2236 005436 000177 173642      2$:     JMP      @OLDPC      ;RETURN TO THE TEST
  
```

```

.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.
*
*****
  
```

```

2249 005442 012637 001304      UBCOUNT:MOV (KSP)+,OLDPC ;SAVE RETURN ADDRESS IN CASE OF ERROR LOOP
2250 005446 050137 001236      BIS      R1,PATTOR      ;LOGICAL OR OF COUNT LOADED
2251 005452 005101      COM      R1              ;GET R1 READY FOR AND
2252 005454 040137 001234      BIC      R1,PATAND      ;PERFORM LOGICAL AND
2253 005460 005101      COM      R1              ;PUT R1 BACK AS IT WAS
2254 005462 050037 001232      BIS      R0,DATAOR      ;LOGICAL OR OF DATA READ
2255 005466 005100      COM      R0              ;GET R0 READY FOR AND
2256 005470 040037 001230      BIC      R0,DATAND      ;PERFORM LOGICAL AND
2257 005474 005100      COM      R0              ;PUT R0 BACK AS IT WAS
2258 005476 005737 001254      TST      ERRCNT          ;IS THIS THE FIRST ERROR?
2259 005502 001002      BNE      1$             ;BRANCH IF NOT FIRST
2260 005504 104205      ERROR   205            ;REPORT FIRST ERROR ON UNIBUS DATA PATH
2261 005506 000401      BR       2$             ;SKIP NEXT INSTRUCTION
2262 005510 104305      1$:     ERROR   305            ;REPORT MORE DATA FROM UNIBUS
2263 005512 000177 173566      2$:     JMP      @OLDPC      ;RETURN TO THE TEST
  
```

```

.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'
*
*****
  
```

2264  
2265  
2266  
2267  
2268  
2269  
2270  
2271  
2272  
2273

L04

```

2274      ;*      PATTERN LOADED INTO THE REGISTERS IN 'PATTOR' AND 'PATAND'.
2275      ;*
2276      ;* *****
2277      005516      CACOUNT:      ;STARTING ADDRESS OF THIS SUBROUTINE
2278      005516      012637      001304      MOV      (KSP)+,OLDPC      ;SAVE RETURN ADDRESS IN CASE OF LOOP
2279      005522      050237      001236      BIS      R2,PATTOR      ;LOGICAL OR OF PATTERN LOADED
2280      005526      005102      COM      R2      ;GET R2 READY FOR AND
2281      005530      040237      001234      BIC      R2,PATAND      ;PERFORM LOGICAL AND
2282      005534      005102      COM      R2      ;PUT R2 BACK AS IT WAS
2283      005536      050337      001232      BIS      R3,DATAOR      ;LOGICAL OR OF DATA FETCHED
2284      005542      005103      COM      R3      ;GET R3 READY FOR AND
2285      005544      040337      001230      BIC      R3,DATAND      ;PERFORM LOGICAL AND
2286      005550      005103      COM      R3      ;PUT R3 BACK AS IT WAS
2287      005552      105737      001103      TSTB     $ERFLG      ;SEE IF THIS IS THE FIRST ERROR
2288      005556      001002      BNE     1$      ;BRANCH IF NOT FIRST ERROR
2289      005560      104207      ERROR     207
2290      005562      000401      BR      2$      ;BRANCH TO EXIT
2291      005564      104307      1$:      ERROR     307
2292      005566      000177      173512      2$:      JMP      @OLDPC      ;RETURN TO TEST
2293
2294
2295      .SBTTL ***** TRAP HANDLING ROUTINES *****
2296
2297      .SBTTL CPU TRAP HANDLER ROUTINE
2298      ;* *****
2299      ;*
2300      ;* THIS SUBROUTINE WILL HANDLE ALL CPU TRAPS AND ABORTS, THRU
2301      ;* "ERRVEC" (000004). IF THIS SUBROUTINE IS ENTERED BY A SECOND
2302      ;* TRAP BEFORE THE FIRST HAS BEEN PROCESSED A HALT IS EXECUTED.
2303      ;* IF THE WORD 'CPUEXP' IS ZERO, NO TRAP WAS EXPECTED AND AN
2304      ;* UNEXPECTED ERROR MESSAGE IS GIVEN. IF THE WORD 'CPUEXP' IS
2305      ;* NOT ZERO THEN THE CPU ERROR REGISTER 'CPUERR' IS COMPARED WITH
2306      ;* 'CPUEXP' TO SEE IF THE PROPER CONDITION OCCURRED. 'PCPUER' CAN
2307      ;* BE USED AS A FLAG TO INDICATE THAT A TRAP HAS OCCURRED SINCE IT
2308      ;* IS LOADED WITH THE ERROR REGISTER IF A TRAP VECTORS HERE
2309      ;* *****
2310      ;* *****
2311      005572      005227      CPUER:      INC      (PC)+      ;MAKE FLAG ZERO IF FIRST TIME
2312      005574      177777      CPFLAG:    .WORD     -1      ;NEGATIVE ONE FOR A FLAG
2313      005576      001401      BEQ      10$      ;BRANCH IF FIRST TIME IN
2314      005600      000000      HALT      ;I HAVE ENTERED THIS ROUTINE BEFORE
2315      ;* I FINISHED REPORTING THE FIRST ERROR
2316      ;* THE SECOND ENTRY ADDRESS IS ON THE
2317      ;* STACK AND THE FIRST ERROR CONDITION
2318      ;* IS PROBABLY STILL LOCKED UP
2319      005602      012637      001304      10$:      MOV      (KSP)+,OLDPC      ;SAVE RETURN ADDRESS IN CASE OF LOOP
2320      005606      012637      001306      MOV      (KSP)+,OLDPS      ;SAVE OLD PSW IN CASE OF LOOP
2321      005612      012706      001100      MOV      #KERSTK,KSP      ;MAKE SURE THAT THE KERNEL STACK POINTER
2322      ;* IS AT 1100.
2323      005616      013737      177766      001266      MOV      CPUERR,PCPUER      ;SAVE CPU ERROR REGISTER
2324      005624      013737      001304      001302      MOV      OLDPC,BADPC      ;SAVE PC+2 AT TIME OF ABORT
2325      005632      005737      001264      TST      CPUEXP      ;SEE IF ANY CONDITION WAS EXPECTED
2326      005636      001406      BEQ      2$      ;BRANCH IF NO TRAP WAS EXPECTED
2327      005640      023737      001266      001264      CMP      PCPUER,CPUEXP      ;SEE IF EXPECTED ERROR OCCURED
2328      005646      001403      BEQ      1$      ;BRANCH IF ERROR CODES MATCH
2329      005650      104001      ERROR     1      ;WRONG CPU TRAP CONDITION
  
```

```

2330 005652 000401 BR 1$ ;SKIP NEXT INSTRUCTION
2331 005654 104002 2$: ERROR 2 ;NO CPU TRAP EXPECTED
2332 005656 013737 001266 177766 1$: MOV PCPUER,CPUERR ;CLEAR CPU ERROR REGISTER
2333 005664 012737 177777 005574 MOV #-1,CPFLAG ;MAKE FLAG NEGATIVE ONE FOR NEXT TIME
2334 005672 013746 001306 MOV OLDPS,-(KSP) ;PUSH OLD PSW BACK ON STACK
2335 005676 013746 001304 MOV OLDPC,-(KSP) ;PUSH RETURN ADDRESS BACK ON STACK
2336 005702 000006 RTT ;RETURN FROM INTERRUPT OR ABORT

```

.SBTTL CACHE TRAPS AND ABOPTS 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.
*
*****

```

```

2353 005704 005227 MEMER: INC (PC)+ ;MAKE FLAG ZERO IF FIRST TIME
2354 005706 177777 PAFLAG: .WORD -1 ;NEGATIVE ONE FOR A FLAG
2355 005710 001401 BEQ 10$ ;BRANCH IF FIRST TIME IN
2356 005712 000000 HALT ;I HAVE ENTERED THIS ROUTINE BEFORE
;I FINISHED REPORTING THE FIRST ERROR
;THE SECOND ENTRY ADDRESS IS ON THE
;STACK AND THE FIRST ERROR CONDITION
;IS PROBABLY STILL LOCKED UP
2361 005714 012737 006126 000114 10$: MOV #5$,CACHVEC ;SET CACHE VECTOR TO RTI UNTIL THE
;THE CACHE HAS BEEN FIXED.
2363 005722 012637 001304 MOV (KSP)+,OLDPC ;SAVE RETURN ADDRESS IN CASE OF LOOP
2364 005726 012637 001306 MOV (KSP)+,OLDPS ;SAVE OLD PSW IN CASE OF LOOP
2365 005732 013737 177740 001270 MOV @#LOADRS, PLOADR ;SAVE LOWER CACHE ADDR REG
2366 005740 013737 177742 001272 MOV @#HIADRS, PHIADR ;SAVE HIGH BITS OF FAILING ADDR
2367 005746 013737 177744 001274 MOV @#MEMERR, PPARER ;SAVE MEMORY ERROR REGISTER
2368 005754 013737 177746 001276 MOV @#CONTRL, PCONTR ;SAVE CONTROL REGISTER FOR TYPE OUT
2369 005762 013737 177750 001300 MOV @#MAINT, PMAINT ;SAVE MAINTENANCE REGISTER
2370 005770 013737 001304 001302 MOV OLDPC, BADPC ;SAVE PC+2 AT TIME OF ABORT
2371 005776 032737 000014 001274 BIT #14, PPARER ;WAS THIS A MAIN MEMORY REFERENCE
2372 006004 001005 BNE 1$ ;BRANCH IF MAIN MEMORY PARITY ERROR
2373 006006 012737 005704 000114 MOV #MEMER, CACHVEC ;LOAD ADDRESS OF THIS ROUTINE IN VECTOR
2374 006014 104003 ERROR 3 ;UNEXPECTED PARITY ABORT
2375 006016 000415 BR 2$ ;BRANCH TO SUBROUTINE EXIT
2376 006020 010046 1$: MOV RO, -(KSP) ;SAVE RO ON STACK
2377 006022 013700 001270 MOV PLOADR, RO ;PUT LOW 16 BITS OF BAD ADDR IN RO
2378 006026 042700 176000 BIC #176000, RO ;CLEAR UPPER 6 BITS OF ADDRESS
2379 006032 074027 000002 XOR RO, #BIT1 ;REFERENCE OTHER WORD OF PAIR
2380 006036 005710 TST (RO) ;READ AT SAME INDEX AS BAD WORD
2381 006040 012600 MOV (KSP)+, RO ;RESTORE RO FROM STACK
2382 006042 012737 005704 000114 MOV #MEMER, CACHVEC ;LOAD ADDRESS OF THIS ROUTINE IN VECTOR
2383 006050 104004 ERROR 4 ;MAIN MEMORY PARITY ERROR
2384 006052 005737 001322 2$: TST RETRY ;ARE YOU RETRYING THIS TEST?
2385 006056 001006 BNE 3$ ;BRANCH IF THIS IS SECOND TRY

```

```

2386 006060 005237 001322      INC      RETRY      ;SET RETRY FLAG
2387 006064 013737 001106 001304  MOV      $LPADR,OLDPC ;RETURN TO START OF THIS TEST
2388 006072 000403      BR       4$        ;BRANCH TO EXIT
2389 006074 013737 001324 001304 3$:  MOV      NXTTST,OLDPC ;RETURN TO START OF NEXT TEST
2390                                ;SINCE THIS IS THE SECOND ABORT
2391 006102 013737 001274 177744 4$:  MOV      PPARER, MEMERR ;CLEAR MEMORY ERROR REGISTER
2392 006110 012737 177777 005706  MOV      #-1,PAFLAG  ;RESTORE A NEGATIVE ONE FOR NEXT TIME
2393 006116 013746 001306      MOV      OLDPS,-(KSP) ;PUSH OLD PSW BACK ON STACK
2394 006122 013746 001304      MOV      OLDPC,-(KSP) ;PUSH RETURN ADDRESS BACK ON STACK
2395 006126 000006      5$:  RTT          ;RETURN FROM INTERRUPT.

```

```

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

```

```

2408 006130 005227      MMTRAP: INC      (PC)+      ;MAKE FLAG ZERO IF FIRST TIME
2409 006132 177777      MMFLAG: .WORD   -1        ;FLAG SHOULD BE NEG ONE
2410 006134 001401      BEQ      10$        ;BRANCH IF FIRST TIME INTO ROUTINE
2411 006136 000000      HALT          ;I HAVE ENTERED THIS ROUTINE BEFORE
2412                                ;I FINISHED REPORTING THE FIRST ERROR
2413                                ;THE SECOND ENTRY ADDRESS IS ON THE
2414                                ;STACK AND THE FIRST ERROR CONDITION
2415                                ;IS PROBABLY STILL LOCKED UP
2416 006140 011637 001302 10$:  MOV      (KSP),BADPC  ;SAVE PC AT TIME OF ABORT OR TRAP
2417 006144 012637 001304  MOV      (KSP)+,OLDPC ;SAVE RETURN ADDRESS IN CASE OF LOOP
2418 006150 012637 001306  MOV      (KSP)+,OLDPS ;SAVE OLD PSW IN CASE OF LOOP
2419 006154 013737 177572 001312  MOV      MMRO,PMRO   ;SAVE STATUS REGISTER
2420 006162 013737 177574 001314  MOV      MMR1,PMR1   ;SAVE AUTO INC/DEC REGISTER
2421 006170 013737 177576 001316  MOV      MMR2,PMR2   ;SAVE VIRTUAL ADDRESS REGISTER
2422 006176 104005      ERROR      5        ;UNEXPECTED M.M. ABORT OR TRAP
2423 006200 042737 177776 177572 1$:  BIC      #177776,@MMRO ;CLEAR ALL BITS EXCEPT 0
2424 006206 012737 177777 006132  MOV      #-1,MMFLAG  ;RESTORE A NEGATIVE ONE TO FLAG
2425 006214 013746 001306  MOV      OLDPS,-(KSP) ;PUSH OLD PSW ONTO STACK
2426 006220 013746 001304  MOV      OLDPC,-(KSP) ;PUSH RETURN ADDRESS ON STACK
2427 006224 000006      RTT          ;RETURN TO MAIN PROGRAM

```

```

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

```

```

2436 010000      START:
2437 010000 012737 000340 177776  MOV      #340,@#PS   ;;LOCK OUT ALL INTERRUPTS
2438 010006 012706 001100  MOV      #$CMTAG,R6 ;FIRST LOCATION TO BE CLEARED
2439 010012 005026      CLR      (R6)+      ;CLEAR MEMORY LOCATION
2440 010014 022706 001136  CMP      #$TKS,R6   ;DONE?
2441 010020 001374      BNE     .-6        ;;LOOP BACK IF NO

```

```

010022 012706 001100 MOV #STACK,SP ;:SETUP THE STACK POINTER
010024 012737 002140 MOV #SCOPE,#IOTVEC ;:IOT VECTOR FOR SCOPE ROUTINE
010026 012737 000340 MOV #340,#IOTVEC+2 ;:LEVEL 7
010028 012737 002460 MOV #ERROR,#EMTVEC ;:EMT VECTOR FOR ERROR ROUTINE
010030 012737 000340 MOV #340,#EMTVEC+2 ;:LEVEL 7
010032 012737 004426 MOV #STRAP,#TRAPVEC ;:TRAP VECTOR FOR TRAP CAL.S
010034 012737 000340 MOV #340,#TRAPVEC+2 ;:LEVEL 7
010036 012737 004470 MOV #SPWRDN,#PWRVEC ;:POWER FAILURE VECTOR
010038 012737 000340 MOV #340,#PWRVEC+2 ;:LEVEL 7
010040 012737 000340 MOV #SENDCT,#SEOPCT ;:SETUP END-OF-PROGRAM COUNTER
010042 012737 005037 CLR #TIMES ;:INITIALIZE NUMBER OF ITERATIONS
010044 012737 005037 CLR #ESCAPE ;:CLEAR THE ESCAPE ON ERROR ADDRESS
010046 012737 000001 MOV #1,#ERRMAX ;:ALLOW ONE ERROR PER TEST
010048 012737 023146 MOV #SRTN,#TBITVEC ;:SET "T" BIT VECTOR TO SRTN
010050 012737 000340 MOV #340,#TBITVEC+2 ;:LEVEL 7
010052 012737 000002 MOV #RTI,#SRTN ;:SET SRTN TO A RTI
010054 012737 010202 MOV #655,#RESVEC ;:TRY TO DO A RTT
010056 012737 005046 CLR #(-SP) ;:DUMMY PS
010058 012746 010172 MOV #645,#(-SP) ;:AND PC
010060 000006 RTT ;:TRY THE RTT
010062 012737 000006 MOV #RTT,#SRTN ;:RTT IS LEGAL--SET SRTN TO A RTT
010064 000402 BR 665 ;:
010066 062706 000010 ADD #10,SP ;:RTT ILLEGAL--CLEAN OFF THE STACK
010068 012737 000012 MOV #RESVEC+2,#RESVEC ;:RESTORE TRAP CATCHER
010070 012737 023154 CLR #TBIT ;:CLEAR "T" BIT SWITCH
010072 012737 010220 MOV #.SLPADR ;:INITIALIZE THE LOOP ADDRESS FOR SCOPE
010074 012737 010226 MOV #.SLPERR ;:SETUP THE ERROR LOOP ADDRESS
010076 005227 INC #-1 ;:FIRST TIME
010078 001042 BNE #E7$ ;:BRANCH IF NO
010080 022737 023100 CMP #SENDAD,#42 ;:ACT-11?
010082 001436 BEQ #67$ ;:BRANCH IF YES
010084 104400 TYPE #68$ ;:TYPE ASCIZ STRING
010086 000433 BR #67$ ;:GET OVER THE ASCIZ
;:69$: .ASCIZ <CRLF>"MAINDEC-11-DEKBF-B PDP-11 70 UNIBUS MAP DIAGNOSTIC CRLF"
;:70$:
;:71$:
;:72$:
;:73$:
;:74$:
;:75$:
;:76$:
;:77$:
;:78$:
;:79$:
;:80$:
;:81$:
;:82$:
;:83$:
;:84$:
;:85$:
;:86$:
;:87$:
;:88$:
;:89$:
;:90$:
;:91$:
;:92$:
;:93$:
;:94$:
;:95$:
;:96$:
;:97$:
;:98$:
;:99$:
010420 005037 001264 CLR #CPUEXP ;:NOT EXPECTING ANY CPU ERRORS
010422 005037 177572 CLR #MMR0 ;:START IN 16 BIT MAPPING
010424 005037 172516 CLR #MMR3 ;:DISABLE MAP AND 22-BIT MAPPING
010426 012700 177777 MOV #-1,RO ;:NEGATIVE ONE USED TO INITIALIZE FLAGS
010428 010037 005574 MOV #RO,#CPFLAG ;:INITIALIZE FLAGS
010430 010037 005074 MOV #RO,#TOFLAG ;:INITIALIZE FLAGS
010432 010037 005706 MOV #RO,#PAFLAG ;:INITIALIZE FLAGS
010434 010037 006132 MOV #RO,#MMFLAG ;:INITIALIZE FLAGS
010436 010037 177744 MOV #RO,#MEMERR ;:CLEAR PARITY ERROR REGISTER

```



2508 010464 010037 177766 MOV R0,0#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:
010470 000004 SCOPE
010470 012737 010554 001324 MOV #TST2,NXTTST ;SAVE STARTING ADDRESS OF NEXT TEST
010472 012737 005072 000004 20\$: MOV #TIMEOUT,ERRVEC ;FOR ESCAPE ON PARITY ERRORS
010500 012737 170200 MOV #MAPLO,R0 ;LOAD ERRVEC WITH ROUTINE ADDR.
010506 012700 010520 001110 MOV #1\$,SLPERR ;PUT FIRST MAP REG ADDR IN R0
010512 012737 010520 001110 1\$: MOV #1\$,SLPERR ;SET LOOP ON ERROR POINTER TO 1\$
010520 000240 NOP ;THIS IS A SYNC POINT FOR SCOPING
010522 011002 MOV (R0),R2 ;READ MAP REGISTERS TO R2
010524 062700 000002 ADD #2,R0 ;POINT TO NEXT MAP REGISTER
010527 022700 170376 CMP #MAPH37,R0 ;COMPARE LAST MAP REG ADDR WITH R0
010534 103371 BHS 1\$ ;CONTINUE READING IF NOT AT LAST REG.
010536 012737 010500 001110 MOV #20\$,SLPERR ;INITIALIZE LOOPING POINTER IN CASE
;OF INTERMITTENT FAULTS.
010544 005737 001254 TST ERRCNT ;SEE IF THERE WERE ANY ERRORS
010550 001401 BEQ TST2 ;NO ERRORS GO TO NEXT TEST
010552 104006 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 IS ON 'MAPB' AND GENERATES 'MAPB CACHE REG'.

\*\*\*\*\*

TST2:
010554 000004 SCOPE
010554 012737 010632 001324 MOV #TST3,NXTTST ;SAVE STARTING ADDRESS OF NEXT TEST
010556 012737 010632 001324 20\$: MOV #LOADRS,R0 ;FOR ESCAPE ON PARITY ERRORS
010564 012700 177740 ;PUT ADDR OF FIRST CACHE REG IN R0



```

2554 010570 012737 010576 001110      MOV      #1$, $LPERR      ;SET LOOP ON ERROR POINTER HERE
2555 010576 000240      NOP                      ;THIS IS A SYNC POINT FOR SCOPING
2556 010600 011002      MOV      (R0), R2        ;TRY TO READ ALL CACHE REGISTERS
2557 010602 062700 000002      ADD      #2, R0          ;POINT TO NEXT CACHE REGISTER
2558 010606 022700 177752      CMP      #HITMIS, R0     ;HAVE ALL REGISTERS BEEN REFERENCED?
2559 010612 103371      BHIS    1$              ;BRANCH IF MORE TO TEST
2560 010614 012737 010564 001110      MOV      #20$, $LPERR   ;INITIALIZE LOOPING ERROR IN CASE
2561                                ;OF INTERMITTENT FAULTS
2562 010622 005737 001254      TST     ERRCNT          ;WERE THERE ANY ERRORS.
2563 010626 001401      BEQ     TST3           ;BRANCH IF NO ERRORS
2564 010630 104007      ERROR   7              ;SUMMARY OF CACHE REGISTERS THAT TIMED OUT
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609

```

```

; *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 MAPLOO - 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 MAPLOO - MAPL17 WILL
; *      GIVE MORE HELP.
; *****

```

```

2587 010632      TST3:
2588 010632 000004      SCOPE
2589 010634 012737 010730 001324      MOV      #TST4, NXTTST   ;SAVE STARTING ADDRESS OF NEXT TEST
2590                                ;FOR ESCAPE ON PARITY ERRORS
2591 010642 012737 005572 000004      MOV      #CPUER, ERRVEC  ;LOAD CPU TRAP SERVICE ROUTINE ADDR
2592 010650 012700 170200      MOV      #MAPLO, R0      ;LOAD STARTING MAP REGISTER ADDR
2593 010654 012737 010662 001110      MOV      #1$, $LPERR    ;SET LOOP ON ERROR POINTER HERE
2594 010662 000240      NOP                      ;THIS IS A SYNC POINT FOR SCOPING
2595 010664 005010      CLR      (R0)           ;CLEAR MAP REGISTER
2596 010666 011037 001170      MOV      (R0), $TMPD     ;SEE IF MAP REGISTER IS ZERO
2597 010672 001402      BEQ     2$              ;BRANCH IF REGISTER IS ZERO
2598 010674 004737 005212      JSR     PC, WRITEERROR  ;REPORT AND LOG ERROR
2599 010700 062700 000004      ADD      #4, R0          ;POINT TO NEXT MAP REG LOWER 16 BITS
2600 010704 022700 170274      CMP      #MAPL17, R0     ;SEE IF THIS SECTION IS TESTED
2601 010710 103364      BHIS    1$              ;BRANCH IF NOT
2602 010712 012737 010650 001110      MOV      #20$, $LPERR   ;INITIALIZE LOOPING POINTER IN CASE
2603                                ;OF INTERMITTENT FAULTS.
2604 010720 005737 001254      TST     ERRCNT          ;WERE THERE ANY ERRORS ON THIS TEST
2605 010724 001401      BEQ     TST4           ;BRANCH IF NO ERRORS
2606 010726 104010      ERROR   10            ;SUMMARY OF MAP REGS THAT COULD NOT
2607                                ;HOLD ZERO
2608
2609

```

E05

2610  
2611  
2612  
2613  
2614  
2615  
2616  
2617  
2618  
2619  
2619  
2620  
2621  
2622  
2623  
2624  
2625  
2626  
2627  
2628  
2629  
2630  
2631  
2632  
2633  
2634  
2635  
2636  
2637  
2638  
2639  
2640  
2641  
2642  
2643  
2644  
2645  
2646  
2647  
2648  
2649  
2650  
2651  
2652  
2653  
2654  
2655  
2656  
2657  
2658  
2659  
2660  
2661  
2662  
2663  
2664  
2665

010730  
010730 000004  
010732 012737 011020 001324  
  
010740 012700 170202  
010744 012737 010752 001110  
010752 000240  
010754 005010  
010756 011037 001170  
010762 001402  
010764 004737 005212  
010770 062700 000004  
010774 022700 170276  
011000 103364  
011002 012737 010740 001110  
  
011010 005737 001254  
011014 001401  
011016 104011

\*\*\*\*\*  
:TEST 4 CLEAR AND READ LOW 20 REGISTERS UPPER 6 BITS  
\*  
\* THIS TEST WILL ENSURE THAT MAP00 - MAP17 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 MAP00 - MAP17 WILL  
\* GIVE MORE HELP.  
\*\*\*\*\*

TST4:  
SCOPE  
MOV #TST5,NXTTST ;SAVE STARTING ADDRESS OF NEXT TEST  
;FOR ESCAPE ON PARITY ERRORS  
20\$: MOV #MAP00,R0 ;LOAD STARTING MAP REGISTER ADDR  
MOV #1\$, \$LPERR ;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 UPPER 6 BITS  
CMP #MAP17,R0 ;SEE IF TESTING IS OVER  
BHS 1\$ ;BRANCH IF MORE REGS TO TEST  
MOV #20\$, \$LPERR ;INITIALIZE LOOPING POINTER IN CASE  
;OF INTERMITTENT FAULTS.  
TST ERRCNT ;WHERE THERE ANY ERRORS ON THIS  
BEQ TST5 ;BRANCH IF NO ERRORS  
ERROR 11 ;SUMMARY OF MAP REGS THAT COULD NOT  
;HOLD ZERO

\*\*\*\*\*  
:TEST 5 CLEAR AND READ HIGH 20 REGISTERS LOWER 16 BITS  
\*  
\* THIS TEST WILL ENSURE THAT MAPL20 - MAPL37 WILL ALL HOLD ZERO.  
\* IF ANY REGISTERS ARE NOT ZERO AFTER BEING CLEARED, THEIR ADDRESS AND  
\* CONTENTS ARE REPORTED. IF ALL THE REGISTERS HAVE RANDOM DATA THEN  
\* PROBABLY NO WRITE PULSE IS BEING GENERATED, BUT IF THE ERROR IS IN  
\* ONLY ONE REGISTER OR IN ONE BIT IT MIGHT BE MORE DIFFICULT TO FIND.  
\* IT IS POSSIBLE THAT THE COUNT PATTERN TEST FOR MAPL20 - MAPL37 WILL  
\* GIVE MORE HELP.  
\*\*\*\*\*

TST5:  
SCOPE  
MOV #TST6,NXTTST ;SAVE STARTING ADDRESS OF NEXT TEST  
;FOR ESCAPE ON PARITY ERRORS  
20\$: MOV #MAPL20,R0 ;LOAD STARTING MAP REGISTER ADDR  
MOV #1\$, \$LPERR ;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

2666	011054	004737	005212		JSR	PC,WRITEERROR	:REPORT AND LOG ERROR
2667	011060	062700	000004	25:	ADD	#4,RO	:POINT TO NEXT REGISTER
2668	011054	022700	170374		CMP	#MAPL37,RO	:SEE IF THIS SECTION IS TESTED
2669	011070	103364			BHIS	15	:BRANCH IF NOT
2670	011072	012737	011030	001110	MOV	#20\$,SLPERR	:INITIALIZE LOOPING POINTER IN CASE
2671							:OF INTERMITTENT FAULTS.
2672	011100	005737	001254		TST	ERRCNT	:WERE THERE ANY ERRORS ON THIS
2673	011104	001401			BEQ	TST6	:BRANCH IF NO ERRORS
2674	011106	104010			ERROR	10	:SUMMARY OF MAP REGS THAT COULD NOT
2675							:HOLD ZERO

```

:*****
: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 BEIN' 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.

```

```

:*****
:TST6:

```

2690	011110						
2691	011110	000004			SCOPE		
2692	011112	012737	011200	001324	MOV	#TST7,NXTTST	:SAVE STARTING ADDRESS OF NEXT TEST
2693							:FOR ESCAPE ON PARITY ERORS
2694	011120	012700	170302		20\$: MOV	#MAPH20,RO	:LOAD STARTING MAP REGISTER ADDR
2695	011124	012737	011132	001110	MOV	#15,\$LPERR	:SET LOOP ON ERROR POINTER HERE
2696	011132	000240			15: NOP		:THIS IS A SYNC POINT FOR SCOPING
2697	011134	005010			CLR	(RO)	:CLEAR MAP REGISTER
2698	011136	011037	001170		MOV	(RO),\$TMPD	:SEE IF MAP REGISTER IS ZERO
2699	011142	001402			BEQ	25	:BRANCH IF REGISTER IS ZERO
2700	011144	004737	005212		JSR	PC,WRITEERROR	:REPORT AND LOG ERROR
2701	011150	062700	000004	25:	ADD	#4,RO	:POINT TO NEXT HIGH REGISTER
2702	011154	022700	170376		CMP	#MAPH37,RO	:SEE IF THERE ARE MORE REGS LEFT
2703	011160	103364			BHIS	15	:BRANCH IF MORE REGS TO TEST
2704	011162	012737	011120	001110	MOV	#20\$,SLPERR	:INITIALIZE LOOPING POINTER IN CASE
2705							:OF INTERMITTENT FAULTS.
2706	011170	005737	001254		TST	ERRCNT	:WERE THERE ANY ERRORS ON THIS TEST
2707	011174	001401			BEQ	TST7	:BRANCH IF NO ERRORS
2708	011176	104011			ERROR	11	:SUMMARY OF MAP REGS THAT COULD NOT
2709							:HOLD ZERO

```

:
: *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
: *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

```

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  
2749  
2750  
2751  
2752  
2753  
2754  
2755  
2756  
2757  
2758  
2759  
2760  
2761  
2762  
2763  
2764  
2765  
2766  
2767  
2768  
2769  
2770  
2771  
2772  
2773  
2774  
2775  
2776  
2777

011200  
011200 000004  
011202 012737 011324 001324  
  
011210 012737 000012 001204  
011216 005003  
011220 005002  
011222 012737 011230 001110  
011230 000240  
011232 010237 170200  
011236 013700 170200  
011242 020200  
011244 001402  
011246 104012  
011250 000404  
  
011252 062702 000002  
011256 001364  
011260 000416  
011262 012737 011270 001110  
011270 000240  
011272 010337 170300  
011276 013701 170300  
011302 020301  
011304 001003  
011306 062703 000002  
011312 001357  
011314 104013  
011316 012737 011216 001110

\*\*\*\*\*  
\* THIS TEST WILL ENSURE THAT THE DATA PATH TO AND FROM THE UNIBUS  
\* MAP REGISTERS IS FUNCTIONING PROPERLY. IT RUNS A COUNT PATTERN  
\* THROUGH MAPLOO, AND IF IT DETECTS AN ERROR THE EXPECTED COUNT  
\* AND THE RECEIVED COUNT ARE REPORTED. THE TEST THEN TRIES TO RUN  
\* THE COUNT THROUGH MAPL20, IN CASE THE ERROR IS IN THE LOWER  
\* GROUP OF REGISTERS AND NOT IN THE DATA PATH.  
\* THE DATA INPUT TO THE MAP REGISTERS IS RECEIVED FROM THE UNIBUS  
\* ON 'MAPA' AND THE OUTPUT TO THE UNIBUS IS DRIVEN ON 'MAPJ'.  
\*\*\*\*\*

\*\*\*\*\*  
TST7:  
SCOPE  
MOV #TST10,NXTTST ;SAVE STARTING ADDRESS OF NEXT TEST  
;FOR ESCAPE ON PARITY ERRORS  
MOV #12,\$TIMES ;DO 12 ITERATIONS  
20\$: CLR R3 ;START COUNT PATTERN AT 0  
CLR R2 ;START COUNT PATTERN AT 0  
MOV #1\$, \$LPERR ;SET LOOP ON ERROR POINTER TO 1\$  
1\$: NOP ;THIS IS A SYNC POINT FOR SCOPING  
MOV R2,MAPLOO ;LOAD MAP REG 0 LOW 16 BITS  
MOV MAPLOO,R0 ;READ CONTENTS OF MAP REGISTER 0  
CMP R2,R0 ;COMPARE DATA & PATTERN  
BEQ 2\$ ;BRANCH IF DATA DOES NOT MATCH  
ERROR 12 ;POSSIBLE DATA PATH FAILURE  
BR 3\$ ;BRANCH TO TEST DATA PATH WITH  
;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\$, \$LPERR ;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\$, \$LPERR ;SET LOOP POINTER TO START OF TEST

\*\*\*\*\*  
\* TEST 10 DATA PATH TEST TO MAP REGISTER UPPER 6 BITS  
\*\*\*\*\*  
\* THIS TEST WILL ENSURE THAT THE DATA PATH TO AND FROM THE UNIBUS  
\* MAP REGISTERS IS FUNCTIONING PROPERLY. IT RUNS A COUNT PATTERN  
\* THROUGH MAPH00, AND IF IT DETECTS AN ERROR THE EXPECTED COUNT  
\* AND THE RECEIVED COUNT ARE REPORTED. THE TEST THEN TRIES TO RUN  
\* THE COUNT THROUGH MAPH20, IN CASE THE ERROR IS IN THE LOWER  
\* GROUP OF REGISTERS AND NOT IN THE DATA PATH.  
\* THE DATA INPUT TO THE MAP REGISTERS IS RECEIVED FROM THE UNIBUS  
\* ON 'MAPA' AND THE OUTPUT TO THE UNIBUS IS DRIVEN ON 'MAPJ'.  
\*\*\*\*\*

011324

TST10:



```

2834 011506 000240          NOP          ; THIS IS A SYNC POINT FOR SCOPING
2835 011510 012710 177777  MOV      #177777, (R0) ; LOAD MAP REGISTER UNDER TEST
2836 011514 011102          2$: MOV      (R1), R2    ; READ MAP REGS STARTING FROM TOP
2837 011516 001404          BEQ      3$         ; GO TO 3$ IF ZERO
2838 011520 020100          CMP      R1, R0     ; SEE IF NON-ZERO ADDRS MATCH
2839 011522 001402          BEQ      3$         ; GO TO 3$ IF MATCH
2840 011524 004737 005276  JSR      PC, DUALADR ; JUMP TO SUBROUTINE TO LOG ALL ERRORS
2841 011530 162701 000002  3$: SUB      #2, R1    ; POINT TO NEXT MAP REGISTER
2842 011534 022701 170200  CMP      #MAPLO, R1 ; HAVE ALL REGISTERS BEEN READ
2843 011540 101765          BLOS    2$         ; IF NOT CONTINUE TEST
2844 011542 062700 000002  ADD      #2, R0     ; POINT TO NEXT MAP REGISTER
2845 011546 022700 170376  CMP      #MAPH37, R0 ; SEE IF TEST IS OVER
2846 011552 103351          BHS     1$         ; CONTINUE TESTING
2847 011554 012737 011464 001110 MOV      #20$, $LPERR ; INITIALIZE LOOPING POINTER IN CASE
2848                                ; OF INTERMITTENT FAULTS.
2849 011562 005737 001254  TST      ERRCNT    ; SEE IF THERE WERE ANY ERRORS
2850 011566 001401          BEQ     TST12     ; GOTO NEXT TEST IF NO ERRORS
2851 011570 104014          ERROR   14       ; SUMMARY DUAL ADDRESSING ERRORS
2852                                ; ON LOADI . 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'.  
\*\*\*\*\*

TST12:

```

2870 011572          TST12:
2871 011572 000004          SCOPE
2872 011574 012737 011716 001324 MOV      #TST13, NXTTST ; SAVE STARTING ADDRESS OF NEXT TEST
2873                                ; FOR ESCAPE ON PARITY ERRORS
2874 011602 012737 000012 001204 MOV      #12, $TIMES ; DO 12 ITERATIONS
2875 011610 005002          20$: CLR      R2        ; START WITH AN ALL ZERO PATTERN
2876 011612 012737 011632 001110 MOV      #2$, $LPERR ; SET LOOP ON ERROR POINTER TO 2$
2877 011620 010204          1$: MOV      R2, R4    ; PUT COUNT IN R4 FOR MASK
2878 011622 042704 000001  BIC      #000001, R4 ; CLEAR BITS IN MASK FOR PROPER COMPARE
2879 011626 012700 170200  MOV      #MAPLOO, R0 ; LOAD STARTING MAP REGISTER ADDRESS
2880 011632 000240          2$: NOP          ; THIS IS A SYNC POINT FOR SCOPING
2881 011634 010210          MOV      R2, (R0)  ; LOAD MAP REGISTER WITH COUNT PATTERN
2882 011636 011003          MOV      (R0), R3 ; READ MAP REGISTER INTO R3
2883 011640 020403          CMP      R4, R3   ; COMPARE MASK WITH DATA
2884 011642 001402          BEQ     3$        ; BRANCH IF DATA MATCHES
2885 011644 004737 005352  JSR      PC, COUNT ; JUMP TO COUNT TO LOG ALL ERRORS
2886 011650 005003          3$: CLR      R3        ; CLEAR DATA HOLDER
2887 011652 062700 000004  ADD      #4, R0   ; POINT TO NEXT MAP REGISTER UNDER TEST
2888 011656 022700 170274  CMP      #MAPL17, R0 ; SEE IF ALL REGISTERS HAVE BEEN LOADED
2889 011662 103363          BHS     2$        ; BRANCH IF STILL MORE TO GO

```





2946  
2947  
2948  
2949  
2950  
2951  
2952  
2953  
2954  
2955  
2956  
2957  
2958  
2959 012042  
2960 012042 000004  
2961 012044 012737 012166 001324  
2962  
2963 012052 012737 000012 001204  
2964 012060 005002 20\$:  
2965 012062 012737 012102 001110  
2966 012070 010204 1\$:  
2967 012072 042704 000001  
2968 012076 012700 170300  
2969 012102 000240 2\$:  
2970 012104 010210  
2971 012106 011003  
2972 012110 020403  
2973 012112 001402  
2974 012114 004737 005352  
2975 012120 005003 3\$:  
2976 012122 062700 000004  
2977 012126 022700 170374  
2978 012132 103363  
2979 012134 022702 177777  
2980 012140 001403  
2981 012142 062702 000401  
2982 012146 000750  
2983 012150 012737 012060 001110 4\$:  
2984  
2985 012156 005737 001254  
2986 012162 001401  
2987 012164 104015  
2988  
2989  
2990  
2991  
2992  
2993  
2994  
2995  
2996  
2997  
2998  
2999  
3000  
3001

\*\*\*\*\*  
\*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'.  
\*\*\*\*\*

TST14:  
SCOPE  
MOV #TST15,NXTTST ;SAVE STARTING ADDRESS OF NEXT TEST  
;FOR ESCAPE ON PARITY ERRORS  
MOV #12,\$TIMES ;DO 12 ITERATIONS  
20\$: CLR R2 ;START WITH AN ALL ZERO PATTERN  
MOV #2\$, \$LPERR ;SET LOOP ON ERROR POINTER TO 2\$  
1\$: MOV R2,R4 ;PUT COUNT IN R4 FOR MASK  
BIC #000001,R4 ;CLEAR BITS IN MASK FOR PROPER COMPARE  
MOV #MAPL20,R0 ;LOAD STARTING MAP REGISTER ADDRESS  
2\$: NOP ;THIS IS A SYNC POINT FOR SCOPING  
MOV R2,(R0) ;LOAD MAP REGISTER WITH COUNT PATTERN  
MOV (R0),R3 ;READ MAP REGISTER INTO R3  
CMP R4,R3 ;COMPARE MASK WITH DATA  
BEQ 3\$ ;BRANCH IF DATA MATCHES  
JSR PC,COUNT ;JUMP TO COUNT TO LOG ALL ERRORS  
3\$: CLR R3 ;CLEAR DATA HOLDER  
ADD #4,R0 ;POINT TO NEXT MAP REGISTER UNDER TEST  
CMP #MAPL37,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  
4\$: MOV #20\$, \$LPERR ;INITIALIZE LOOPING POINTER IN CASE  
;OF INTERMITTENT FAULTS.  
TST ERRCNT ;SEE IF THERE WERE ANY ERRORS  
BEQ TST15 ;BRANCH IF NO ERRORS  
ERROR 15 ;SUMMARY OF COUNT PATTERN FAILURES  
;IN MAPPING REGISTERS

\*\*\*\*\*  
\*TEST 15 COUNT PATTERN HIGH 20 MAP REGISTERS UPPER 6 BITS  
\*

\* THIS TEST WILL RUN A COUNT PATTERN THROUGH UNIBUS MAP REGISTERS  
\* MAPH20 - MAPH37. IF THE COUNT 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'.  
\*\*\*\*\*



L05

```

3002 .....
3003 012166 TST15:
3004 012166 000004 SCOPE
3005 012170 012737 012312 001324 MOV #TST16,NXTTST ;SAVE STARTING ADDRESS OF NEXT TEST
3006 ;FOR ESCAPE ON PARITY ERRORS
3007 012176 012737 000144 001204 MOV #144,$TIMES ;DO 144 ITERATIONS
3008 012204 005002 20$: CLR R2 ;START WITH AN ALL ZERO PATTERN
3009 012206 012737 012226 001110 MOV #2$, $LPERR ;SET LOOP ON ERROR POINTER TO 2$
3010 012214 010204 1$: MOV R2,R4 ;PUT COUNT IN R4 FOR MASK
3011 012216 042704 177700 BIC #177700,R4 ;CLEAR BITS IN MASK FOR PROPER COMPARE
3012 012222 012700 170302 MOV #MAPH20,R0 ;LOAD STARTING MAP REGISTER ADDRESS
3013 012226 000240 2$: NOP ;THIS IS A SYNC POINT FOR SCOPING
3014 012230 010210 MOV R2,(R0) ;LOAD MAP REGISTER WITH COUNT PATTERN
3015 012232 011003 MOV (R0),R3 ;READ MAP REGISTER INTO R3
3016 012234 020403 CMP R4,R3 ;COMPARE MASK WITH DATA
3017 012236 001402 BEQ 3$ ;BRANCH IF DATA MATCHES
3018 012240 004737 005352 JSR PC,COUNT ;JUMP TO COUNT TO LOG ALL ERRORS
3019 012244 005003 3$: CLR R3 ;CLEAR DATA HOLDER
3020 012246 062700 000004 ADD #4,R0 ;POINT TO NEXT MAP REGISTER UNDER TEST
3021 012252 022700 170376 CMP #MAPH37,R0 ;SEE IF ALL REGISTERS HAVE BEEN LOADED
3022 012256 103363 BHIS 2$ ;BRANCH IF STILL MORE TO GO
3023 012260 022702 000177 CMP #177,R2 ;SEE IF COUNT HAS CYCLED
3024 012264 001403 BEQ 4$ ;BRANCH IF TEST IS DONE
3025 012266 062702 000001 ADD #1,R2 ;CHANGE COUNT PATTERN
3026 012272 000750 BR 1$ ;CONTINUE TESTING
3027 012274 012737 012204 001110 4$: MOV #20$, $LPERR ;INITIALIZE LOOPING POINTER IN CASE
3028 ;OF INTERMITTENT FAULTS.
3029 012302 005737 001254 TST ERRCNT ;SEE IF THERE WERE ANY ERRORS
3030 012306 001401 BEQ TST16 ;BRANCH IF NO ERRORS
3031 012310 104016 ERROR 16 ;SUMMARY OF COUNT PATTERN FAILURES
3032 ;IN MAPPING REGISTERS

```

```

3033 .....
3034 .....
3035 .....
3036 *TEST 16 CACHE REGISTER DATA PATHS
3037 *
3038 * THIS TEST MAKES AN EFFORT TO VERIFY THE DATA PATH FROM THE
3039 * CACHE REGISTERS THROUGH THE UNIBUS MAP TO THE CPU. IT FIRST
3040 * CHECKS THE CONTROL REGISTER (177746) TO SEE IF THE CACHE IS
3041 * DISABLED. IF IT IS NOT DISABLED THEN THE CONTROL REGISTER
3042 * IS CLEARED AND VERIFIED TO BE ZERO. THE MAINTENANCE REGISTER
3043 * IS ALWAYS CLEARED AND VERIFIED TO BE ZERO. THEN THE CACHE
3044 * ADDRESS REGISTERS (177740 & 177742) ARE READ, WITH THE ERROR
3045 * REGISTER CLEARED. THEY SHOULD BE 177740 AND 000003 RESPECTIVELY.
3046 * IF ANY OF THESE CONDITIONS ARE NOT MET AN ERROR IS REPORTED
3047 * AS BEING A POSSIBLE CACHE REGISTER DATA PATH ERROR.
3048 * THE INPUT TO THE CACHE REGISTERS IS FROM 'MAPA' AND THE CACHE
3049 * DATA MULTIPLEXER IS ON 'MAPH'. THE REGISTER INPUTS ARE THE
3050 * "X" INPPUTS, THESE GO TO 'MAPJ' TO GET BACK TO THE CPU.
3051 *

```

```

3052 .....
3053 012312 TST16:
3054 012312 000004 SCOPE
3055 012314 012737 012502 001324 MOV #TST17,NXTTST ;SAVE STARTING ADDRESS OF NEXT TEST
3056 ;FOR ESCAPE ON PARITY ERRORS
3057 012322 012737 012342 001110 20$: MOV #10$, $LPERR ;SET LOOP ON ERROR POINTER TO 10$

```

M05

```

3058 012330 005000 CLR RO ;CLEAR RO IN CASE CACHE IS DISABLED
3059 012332 032737 000014 177746 BIT #14,2#CONTRL ;SEE IF EITHER GROUP IS DISABLED
3060 012340 001007 BNE 1$ ;BRANCH IF CACHE IS DISABLED
3061 012342 000240 10$: NOP ;THIS IS A SYNC POINT FOR SCOPING
3062 012344 005037 177746 CLR 2#CONTRL ;CLEAR CONTROL REGISTER
3063 012350 013700 177746 MOV 2#CONTRL,RO ;READ CONTROL REGISTER
3064 012354 001401 9EQ 1$ ;BRANCH IF CONTROL REGISTER IS ZERO
3065 012356 104017 ERROR 17 ;POSSIBLE FAULT IN CACHE REG DATA PATHS
3066 012360 012737 012366 001110 1$: MOV #11$, $LPERR ;SET LOOP ON ERROR POINTER TO 11$
3067 012366 000240 11$: NOP ;THIS IS A SYNC POINT FOR SCOPING
3068 012370 005037 177750 CLR 2#MAINT ;CLEAR MAINTENANCE REGISTER
3069 012374 013701 177750 MOV 2#MAINT,R1 ;READ MAINTENANCE REGISTER
3070 012400 001401 BEQ 2$ ;BRANCH IF MAINTENANCE REGISTER IS ZERO
3071 012402 104020 ERROR 20 ;MAINTENANCE REGISTER WILL NOT CLEAR
3072 012404 012737 012412 001110 2$: MOV #12$, $LPERR ;SET LOOP ON ERROR POINTER TO 12$
3073 012412 012737 177777 177744 12$: MOV #-1, MEMERR ;MAKE SURE THAT CACHE ERROR REG IS CLEAR
3074 012420 000240 NOP ;THIS IS A SYNC POINT FOR SCOPING
3075 012422 013700 177740 MOV LOADRS,RO ;READ CACHE LOW ADDR REG TO RO
3076 012426 022700 177740 CMP #177740,RO ;SEE IF RO = ADDR OF CACHE LOW ADDR REG
3077 012432 001401 BEQ 3$ ;BRANCH IF DATA READ CORRECTLY
3078 012434 104021 ERROR 21 ;CANNOT READ 177740 FROM "LOADRS"
3079 012436 012737 012444 001110 3$: MOV #13$, $LPERR ;SET LOOP ON ERROR POINTER TO 13$
3080 012444 012737 177777 177744 13$: MOV #-1, MEMERR ;MAKE SURE THAT CACHE ERROR REG IS CLEAR
3081 012452 000240 NOP ;THIS IS A SYNC POINT FOR SCOPING
3082 012454 013700 177742 MOV HIADRS,RO ;READ CACHE HIGH ADDR REG TO RO
3083 012460 042700 177700 BIC #177700,RO ;MASK OFF UPPER 10 BITS
3084 012464 022700 000003 CMP #3,RO ;SEE IF RO = 3
3085 012470 001401 BEQ 14$ ;BRANCH IF DATA MATCHES
3086 012472 104022 ERROR 22 ;CANNOT READ 000003 FROM "HIADRS"
3087 012474 012737 012322 001110 14$: MOV #20$, $LPERR ;SET LOOP POINTER TO START OF TEST
    
```

```

3088
3089
3090 ;*****
3091 ;TEST 17 CACHE REGISTER DATA PATH COUNT PATTERN
3092 ;
3093 ; THIS TEST FIRST SAVES THE CONDITION OF THE CACHE CONTROL
3094 ; REGISTER IS "$TMP5" SO THAT THE CACHE CAN BE RESTORED.
3095 ; IF EITHER BIT 3 OR 4 IS ON THE TEST BRANCHES TO "3$" TO
3096 ; TEST BITS 5 - 15 OF THE DATA PATH, IF NEITHER BIT IS ON THEN
3097 ; THE TEST RUNS A COUNT PATTERN THRU THE CONTROL REGISTER FROM
3098 ; 00 - 77. IF ANY BITS ARE DETECTED AS BEING STUCK THE ERROR
3099 ; IS REPORTED AND A SUMMARY OF THOSE ERRORS IS GIVEN. THEN THE
3100 ; TEST FORCES MISSES IN BOTH GROUPS OF THE CACHE AND PROCEEDS
3101 ; TO RUN A COUNT PATTERN THROUGH THE CACHE MAINTENANCE REGISTER
3102 ; (177746) FROM 000000 - 177760 IN INCREMENTS OF 20.
3103 ; THE CODE IS SET UP IN SUCH A WAY THAT NO PARITY ERRORS
3104 ; SHOULD BE GENERATED. AGAIN ALL BAD MATCHES OF DATA WILL BE
3105 ; REPORTED AND A SUMMARY OF ALL ERRORS IS GIVEN AT THE END OF
3106 ; THE TEST AND THE CACHE IS RESTORED TO ITS PREVIOUS CONDITION.
3107 ;
3108 ;*****
    
```

```

3109 012502 TST17:
3110 012502 000004 SCOPE
3111 012504 012737 013124 001324 MOV #TST20,NXTTST ;SAVE STARTING ADDRESS OF NEXT TEST
3112 ;FOR ESCAPE ON PARITY ERRORS
3113 012512 012737 000012 001204 MOV #12,$TIMES ;DO 12 ITERATIONS
    
```

3114	012520	104416				TBITO			;TURN OFF T BIT TRAPPING IF ON
3115	012522	013737	177746	001202	20\$:	MOV	2#CONTRL,\$TMP5		;SAVE CONTROL REG TO RESTORE CACHE
3116	012530	032737	000014	177746		BIT	#14,2#CONTRL		;SEE IF EITHER GROUP IS DISABLED
3117	012536	001047				BNE	3\$		;SKIP FIRST PART OF TEST IF DISABLED
3118	012540	012700	177746			MOV	#CONTRL,R0		;LOAD ADDR OF CONTROL REG INTO R0
3119	012544	005002				CLR	R2		;START COUNT PATTERN AT ZERO
3120	012546	012737	012554	001110		MOV	#1\$, \$LPERR		;SET LOOP ON ERROR POINTER TO 1\$
3121	012554	000240			1\$:	NOP			;THIS IS A SYNC POINT FOR SCOPING
3122	012556	010210				MOV	R2,(R0)		;LOAD COUNT INTO CONTROL REG
3123	012560	011003				MOV	(R0),R3		;READ COUNT FROM CONTROL REG
3124	012562	020203				CMP	R2,R3		;SEE IF COUNT = DATA READ
3125	012564	001402				BEQ	2\$		;BRANCH IF DATA MATCHES
3126	012566	004737	005516			JSR	PC,CACOUNT		;LOG AND REPORT COUNT PATTERN ERROR
3127	012572	005202			2\$:	INC	R2		;CHANGE COUNT PATTERN
3128	012574	022702	000077			CMP	#77,R2		;SEE IF COUNT HAS PASSED MAXIMUM
3129	012600	103365				BHIS	1\$		;BRANCH IF COUNT HASN'T CYCLED
3130	012602	005737	001254			TST	ERRCNT		;SEE IF ANY ERRORS ON THIS CYCLE
3131	012606	001423				BEQ	3\$		;BRANCH IF NO ERRORS
3132	012610	104023				ERROR	23		;COUNT THRU CONTROL REG FAILED
3133	012612	005037	001322			CLR	RETRY		;CLEAR RETRY FLAG AN THE START OF
3134									;EACH TEST
3135	012616	005037	001254			CLR	ERRCNT		;CLEAR THE MULTIPLE ERKOR COUNTER
3136	012622	005037	001232			CLR	DATAOR		;LOCATION FOR LOGICAL OR OF BAD DATA
3137	012626	005037	001226			CLR	ADDROR		;LOCATION FOR LOGICAL OR OF ADDRESS
3138	012632	005037	001236			CLR	PATTOR		;LOCATION FOR LOGICAL OR OF PATTERN LOADED
3139	012636	012700	177777			MOV	#-1,R0		;LOAD -1 INTO R0 TO INITIALIZE LOGICAL AND LOCS
3140	012642	010037	001230			MOV	R0,DATAND		;LOCATION FOR LOGICAL AND OF BAD DATA
3141	012646	010037	001224			MOV	R0,ADRAND		;LOCATION FOR LOGICAL AND OF ADDRESS
3142	012652	010037	001234			MOV	R0,PATAND		;LOCATION FOR LOGICAL AND OF PATTERN LOADED
3143	012656	012737	000014	177746	3\$:	MOV	#14,2#CONTRL		;DISABLE BOTH GROUPS OF CACHE
3144	012664	012700	177750			MOV	#MAINT,R0		;LOAD ADDR OF MAINTENENCE REG INTO R0
3145	012670	010001				MOV	R0,R1		;R1 HAS #MAINT SO CLR (R1) HAS P.B.'S ON
3146	012672	005002				CLR	R2		;START COUNT PATTERN AT ZERO
3147	012674	012737	012702	001110		MOV	#4\$, \$LPERR		;SET LOOP ON ERROR POINTER TO 4\$
3148	012702	010205			4\$:	MOV	R2,R5		;PUT COUNT IN R5 FOR CORRECT PARITY
3149	012704	000240				NOP			;THIS IS A SYNC POINT FOR SCOPING
3150	012706	010510				MOV	R5,(R0)		;LOAD MAINT REGISTER INST HAS P.B.'S ON
3151	012710	011003				MOV	(R0),R3		;READ MAINT REG (P.B.'S ON)
3152	012712	005011				CLR	(R1)		;TURN OFF MAINT REG (P.B.'S ON)
3153	012714	050000				BIS	R0,R0		;DUMMY INST WITH P.B.'S ON
3154	012716	020203				CMP	R2,R3		;SEE IF COUNT = DATA READ
3155	012720	001402				BEQ	5\$		;BRANCH IF DATA MATCHES
3156	012722	004737	005516			JSR	PC,CACOUNT		;LOG AND REPORT COUNT PATTERN ERROR
3157	012726	062702	000020		5\$:	ADD	#20,R2		;CHANGE COUNT PATTERN
3158	012732	001363				BNE	4\$		;BRANCH IF COUNT HAS NOT PASSED 177760
3159	012734	005737	001254			TST	ERRCNT		;SEE IF ANY ERRORS ON THIS COUNT CYCLE
3160	012740	001401				BEQ	19\$		;BRANCH IF NO ERRORS ON THIS PART
3161	012742	104024				ERROR	24		;SUMMARY OF COUNT ERRORS IN MAINT REG
3162	012744	012737	012522	001110	19\$:	MOV	#20\$, \$LPEOP		;SET LOOP POINTER TO START OF TEST
3163	012752	013737	001202	177746		MOV	\$TMP5,2#CONTRL		;RETURN CACHE TO PREVIOUS STATE
3164	012760	000004				SCOPE			
3165									
3166									
3167									
3168									
3169									

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

```

012762 104420          TBITR          ;RESTORE THE T BIT TO ITS CONDITION
012764 012700 077406  MOV          #77406,R0      ;BEFORE THE LAST TEST
                                ;MAKE THE KERNEL I-SPACE PAGES ALL
                                ;4K UPWARD EXPANDABLE, READ/WRITE
012773 010037 172300  MOV          R0,KIPOR0      ;KERNEL I-SPACE PAGE 0
012774 010037 172302  MOV          R0,KIFDR1      ;KERNEL I-SPACE PAGE 1
013000 010037 172304  MOV          R0,KIPDR2      ;KERNEL I-SPACE PAGE 2
013004 010037 172306  MOV          R0,KIPDR3      ;KERNEL I-SPACE PAGE 3
013010 010037 172310  MOV          R0,KIPDR4      ;KERNEL I-SPACE PAGE 4
013014 010037 172312  MOV          R0,KIPDR5      ;KERNEL I-SPACE PAGE 5
013020 010037 172314  MOV          R0,KIPDR6      ;KERNEL I-SPACE PAGE 6
013024 010037 172316  MOV          R0,KIPDR7      ;KERNEL I-SPACE PAGE 7
013030 012737 000000  MOV          #000,KIPAR0     ;MAP TO PHYSICAL 0
013034 012737 000200  MOV          #200,KIPAR1     ;MAP TO PHYSICAL 4K - 8K
013044 012737 000400  MOV          #400,KIPAR2     ;MAP TO PHYSICAL 8K - 12K
013050 012737 000600  MOV          #600,KIPAR3     ;MAP TO PHYSICAL 12K - 16K
013060 012737 001000  MOV          #1000,KIPAR4    ;MAP TO PHYSICAL 16K - 20K
013066 012737 001200  MOV          #1200,KIPAR5    ;MAP TO PHYSICAL 20K - 24K
013074 012737 170000  MOV          #17000,KIPAR6   ;MAP TO UNIBUS
013102 012737 177600  MOV          #177600,KIPAR7  ;MAP TO I/O PAGE
013110 012737 000001  MOV          #8BIT0,MMR0     ;ENABLE FULL 18-BIT MAPPING
013116 012737 000020  MOV          #8BIT4,MMR3     ;ENABLE 22-BIT MAPPING

```

\*\*\*\*\*  
 \*TEST 20 MAP REGISTER ADDRESS DECODE TEST  
 \*\*\*\*\*

THIS TEST TRIES TO VERIFY THAT NONE OF THE INPUTS TO THE ADDRESS  
 DECODER FOR THE UNIBUS MAP REGISTERS ON 'MAPB' IS STUCK TRUE.  
 KIPAR6 IS SETUP TO HOLD 177702 AND R4 HAS THE VIRTUAL ADDRESS  
 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.

```

013124 012737 013154 001106 20S: MOV          #20S,$_PADR      ;SET LOOP ON TEST POINTER TO 20S
013124 012737 013154 001110  MOV          #20S,$LPERR     ;SET LOOP ON ERROR POINTER TO 20S
013132 012737 000020 001102  MOV          #20,$STNMM      ;SETUP TEST NUMBER AND CLR ERROR FLAG
013140 013737 001102 177570  MOV          $STNMM,DISPLAY  ;DISPLAY TEST NUMBER FOR ALL TO SEE
                                ;BIT4 IS TIMEOUT BIT IN CPU ERROR REG
                                ;EXPECTING CPU TIME OUT ON UNIBUS
013154 012737 000020 001264 20S: MOV          .EQUIV BIT4,TIMEOUT CPUEXP ;SET LOOP ON ERROR POINTER TO 10S
013162 012737 013222 001110  MOV          #10S,$LPERR     ;SET LOOP ON ERROR POINTER TO 10S
013170 012737 177702 172354  MOV          #177702,KIPAR6  ;PUT MAP REG 0 ADDR IN PAR6
                                ;PATTERN FOR TESTING.
013176 012702 175254      MOV          #175254,R2      ;LOAD MAP REGISTER 0
013202 010237 170200      MOV          R2,$MAPLO       ;SET BIT 11 TO FLOAT THRU PAR6
013206 012700 004000      MOV          #8BIT11,R0      ;VIRT. ADDR. TO SELECT PAR6
013212 012704 140000      MOV          #140000,R4      ;CHANGE A BIT OF MAP REG 0'S ADDR
013216 074037 172354      1S: XOR          R0,KIPAR6     ;THIS IS A SYNC POINT FOR SCOPING
013222 000240      1CS: NOP                          ;READ LOCATION POINTED TO BY PAR6
013224 011401      MOV          (R4),R1         ;SEE IF DATA FETCHED MATCHES PATTERN
013226 020201      CMP          R2,R1           ;BRANCH IF NOT SAME
013230 001007      BNE          JS             ;TRY TO CLEAR THIS LOCATION
013232 012714 000000      MOV          #0,(R4)         ;SEE IF MAP REG 0 GOT CLEARED
013236 005737 170200      TST          $MAPLO         ;BRANCH IF MAP REG NOT ZERO
013242 001001      BNE          JS

```

```

3226 013244 104025          ERROR 25          :GOT TO MAPLO WITH ONE BIT
3227          :DIFFERENT IN ADDRESS FROM 770200
3228 013246 010114          25:  MOV  R1,(R4)      :RELOAD THE LOCATION
3229 013253 074037 172354  35:  XOR  R0,KIPAR6    :RESTORE BIT TO ORIGINAL STATUS
3230 013254 006200          :ASR  R0             :RIGHT SHIFT ONE PLACE
3231 013256 020027 000001  :CMP  R0,#1         :SEE IF TEST IS OVER
3232 013262 001355          :BNE  15            :BRANCH TO CONTINUE TEST
3233 013264 012737 013154 001110 :MOV  #205,$LPERR   :SET LOOP POINTER TO START OF TEST
3234 013272 005037 001264  :CLR  CPUEXP        :ZERO EXPECTED CPU TRAP FLAG

```

\*\*\*\*\*  
:TEST 21 CACHE REGISTER ADDRESS DECODE TEST  
\*\*\*\*\*

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

\*\*\*\*\*  
†ST21:  
\*\*\*\*\*

```

3247 013276          SCOPE
3248 013276 000004          MOV  #TST22,NXTTST  :SAVE STARTING ADDRESS OF NEXT TEST
3249 013300 012737 013500 001324  :FOR ESCAPE ON PARITY ERRORS
3250          :MOV  #TIMOUT,CPUEXP  :EXPECTING UNIBUS TIMEOUT IN THIS TEST
3251 013306 012737 000020 001264 205:  MOV  #105,$LPERR   :SET LOOP ON ERROR POINTER TO 105
3252 013314 012737 013350 001110  :MOV  #177777,KIPAR6 :PUT CACHE REG BASE IN PAR6
3253 013322 012737 177777 172354  :MOV  #177740,R2     :PUT DATA PATTERN IN R2
3254 013330 012702 177740          :MOV  #BITS,R0       :SET BIT 5 TO FLOAT THRU PAR6
3255 013334 012700 000040          :MOV  #140040,R4    :VIRT. ADDR. TO SELECT PAR6
3256 013340 012704 140040          :XOR  R0,KIPAR6     :CHANGE A BIT OF CACHE REG'S ADDR
3257 013344 074037 172354          15:  NOP                :THIS IS A SYNC POINT FOR SCOPING
3258 013350 000240          105: MOV  (R4),R1        :TRY TO READ CACHE 'LOADRS' WITH
3259 013352 011401          :CMP  R2,R1         :ONE BIT DIFFERENT THAN ADDR. 777740
3260          :BNE  25            :SEE IF YOU GOT CACHE 'LOADRS'
3261 013354 020201          :ERROR 26           :BRANCH TO CONTINUE TEST.
3262 013356 001001          :XOR  R0,KIPAR6    :READ CACHE 'LOADRS' WITHOUT ADRS 777740
3263 013360 104026          25:  ASR  R0             :RESTORE BIT TO ORIGINAL STATUS.
3264 013362 074037 172354          :MOV  R0            :RIGHT SHIFT R0 1 PLACE
3265 013366 006200          :BNE  15            :BRANCH IF NOT 0
3266 013370 001365          :MOV  #115,$LPERR  :SET LOOP ON ERROR POINTER TO 115
3267 013372 012737 013404 001110  :MOV  #177700,R1    :PUT ADDRESS IN R1
3268 013400 012701 177700          115: NOP                :THIS IS A SYNC POINT FOR SCOPING
3269 013404 000240          :MOV  (R1),R3       :TRY TO READ 177700
3270 013406 011103          :CMP  R2,R3         :SEE IF YOU GET 177740
3271 013410 020203          :BNE  35            :BRANCH IF YOU DON'T
3272 013412 001001          :ERROR 27           :READ CACHE 'LOADRS' WITHOUT ADRS 777740
3273 013414 104027          35:  MOV  #125,$LPERR  :SET LOOP ON ERROR POINTER TO 125
3274 013416 012737 013430 001110  :MOV  #177760,R1    :PUT ADDRESS IN R1
3275 013424 012701 177760          125: NOP                :THIS IS A SYNC POINT FOR SCOPING
3276 013430 000240          :MOV  (R1),R3       :TRY TO READ 177760
3277 013432 011103          :CMP  R2,R3         :SEE IF YOU GET 177740
3278 013434 020203          :BNE  45            :BRANCH IF YOU DON'T
3279 013436 001001          :ERROR 27           :READ CACHE 'LOADRS' WITHOUT ADRS 777740
3280 013440 104027          45:  MOV  #135,$LPERR  :SET LOOP ON ERROR POINTER TO 135
3281 013442 012737 013454 001110

```

```

3282 013450 012701 177754          MOV      #177754,R1      ;PUT ADDRESS IN R1
3283 013454 000240          13$:    NOP              ;THIS IS A SYNC POINT FOR SCOPING
3284 013456 011103          MOV      (R1),R3       ;TRY TO READ 177754
3285 013460 020203          CMP      R2,R3         ;SEE IF YOU GET 177740
3286 013462 001001          BNE     19$           ;BRANCH IF YOU DIDN'T READ ADDR 177740
3287 013464 104027          ERROR   27           ;READ CACHE 'LOADRS' WITHOLT ACRS 777740
3288 013466 012737 013306 001110 19$:    MOV      #20$,SLPERR   ;SET LOOP POINTER TO START OF TEST
3289 013474 005037 001264          CLR     CPUEXP        ;ZERO EXPECTED CPU TRAP CONDITION
3290
3291
3292
3293
3294
3295
3296
3297
3298
3299
3300
3301
3302
3303
3304
3305
3306
3307
3308
3309 013500          *-----*
3310 013500          *TEST 22      DATA PATH, UNIBUS TO MAIN MEMORY
3311 013502 012737 013732 001324          *
3312          * THIS TEST RUNS A COUNT PATTERN THROUGH A MEMORY LOCATION VIA
3313          * THE UNIBUS.  THE UNIBUS MAP IS LEFT OFF DURING THIS TEST SO
3314          * THAT THE ADDRESS IS NOT RELOCATED.  THE TEST TRIES TO LOAD THE
3315          * PATTERN INTO ADDRESS 040000 (8K) BUT IF THE MAP JUMPERS ARE
3316          * SET NOT TO RESPOND TO THAT ADDRESS THE NEXT 4K IS TRIED UNTIL
3317          * THE TEST GETS TO MAIN MEMORY FROM THE UNIBUS.  IF THIS TEST
3318          * DETERMINES THAT IT CANNOT GET TO MAIN MEMORY FROM THE UNIBUS
3319          * IT REPORTS THE FACT AND SKIPS THE NEXT TEST FOR VERIFICATION.
3320          * THE DATA PATH TO MAIN MEMORY FROM THE UNIBUS IS ON 'MAPA' AND
3321          * THE PATH FROM MAIN MEMORY TO THE UNIBUS IS FROM THE CACHE
3322          * "DTML CDMX D00" - "DTML CDMX D15" INTO 'MAPH' THE 'Z' INPUTS
3323          * TO THE MULTIPLEXER AND THEN TO 'MAPJ' AND OUT TO THE UNIBUS.
3324          *
3325          *-----*
3326          *TST22:
3327          SCOPE
3328 013510 012737 000012 001204          MOV      #TST23,NXTTST ;SAVE STARTING ADDRESS OF NEXT TEST
3329 013516 004737 005054          20$:    JSR      PC,CLAMAP   ;FOR ESCAPE ON PARITY ERRORS
3330 013522 012737 000020 001264          MOV      #12,$TIMES    ;DO 12 ITERATIONS
3331 013530 012737 013536 001110          MOV      #TIMOUT,CPUEXP ;TIMEOUTS MIGHT OCCUR IN THIS TEST.
3332 013544 005037 001266          10$:    MOV      #10$,SLPERR   ;SET LOOP ON ERROR POINTER TO 10$
3333 013550 000240          1$:    CLR     PCPUER       ;CLEAR ERROR CONDITION LOCATION
3334 013552 013700 140000          MOV      @#140000,R0    ;THIS IS A SYNC POINT FOR SCOPING
3335 013556 005737 001266          TST     PCPUER         ;TRY TO READ ADDRESS POINTED TO BY PAR6
3336 013562 001412          BEQ     2$            ;SEE IF READ OF ADDRESS TIMED OUT
3337 013564 062737 000200 172354 4$:    ADD      #200,KIPAR6   ;BRANCH IF REFERENCE WAS GOOD
3338 013572 022737 177600 172354          CMP      #177600,KIPAR6 ;TRY NEXT 4K BLOCK OF MEMORY
3339 013600 001361          BNE     1$           ;SEE IF YOU'VE POINTED TO I/O PAGE
3340 013602 104030          ERROR   30           ;BRANCH IF NOT
3341 013604 000137 014062          JMP     SIZEJ         ;NO UNIBUS ADDRESSES RESPOND
3342 013610 013737 172354 172352 2$:    MOV      KIPAR6,KIPAR5 ;JUMP TO SIZE JUMPER TEST
3343 013616 042737 170000 172352          BIC     #170000,KIPAR5 ;PUT PAR6 INTO PAR5
3344 013624 012737 173214 120000          MOV      #173214,@#120000 ;MAKE PAR5 A NON UNIBUS ADDRESS
3345 013632 013701 140000          MOV      @#140000,R1   ;PUT RANDOM NUMBER INTO TEST LOCATION BY FAST BUS
3346 013636 022701 173214          CMP      #173214,R1    ;READ TEST LOCATION VIA UNIBUS
3347 013642 001403          BEQ     3$           ;SEE IF DATA WAS READ PROPERLY
3348 013644 020001          CMP     R0,R1         ;DATA OKAY NOW VERIFY DATA PATH
3349 013646 001001          BNE     3$           ;SEE IF DATA CHANGED FROM FIRST READ
3350 013650 000745          BR      4$           ;BRANCH IF DATA CHANGED
3351 013652 005001          3$:    CLR     R1         ;TRY NEXT 4K BLOCK
3352          ;CLEAR REGISTER TO HOLD COUNT

```



```

3338 013654 012737 013666 001110      MOV      #11$, $LPERR      ;SET LOOP ON ERROR POINTER TO 11$
3339 013662 012702 140000              MOV      #140000, R2      ;LOAD VIRTUAL ADDRESS INTO R2
3340 013666 000240              11$:    NOP                    ;THIS IS A SYNC POINT FOR SCOPING
3341 013670 010112              5$:    MOV      R1, (R2)      ;LOAD COUNT INTO TEST LOCATION VIA U.B.
3342 013672 011200              MOV      (R2), R0        ;READ TEST LOCATION BACK VIA UNIBUS
3343 013674 020100              CMP      R1, R0          ;COMPARE COUNT WITH DATA READ
3344 013676 001402              BEQ      6$              ;BRANCH IF DATA MATCHES
3345 013700 004737 005442              JSR      PC, UBCOUNT     ;COUNT FAILED REPORT ERROR
3346 013704 005201              6$:    INC      R1          ;INCREASE COUNT
3347 013706 001370              BNE      5$              ;BRANCH IF COUNT HASN'T CYCLED
3348 013710 005737 001254              TST      ERRCNT          ;WERE THERE ANY ERRORS ON THIS TEST
3349 013714 001401              BEQ      19$             ;BRANCH IF NO ERRORS ON THIS TEST
3350 013716 104031              ERROR   31               ;SUMMARY OF ERRORS ON THE UNIBUS
3351                                ;DATA PATH
3352 013720 012737 013516 001110 19$:    MOV      #20$, $LPERR     ;SET LOOP POINTER TO START OF TEST
3353 013726 005037 001264              CLR      CPUEXP          ;ZERO EXPECTED CPU TRAP CONDITION

```

```

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

```

```

3370 013732                                TST23:
3371 013732 000004              SCOPE
3372 013734 012737 014044 001324      MOV      #TST24, NXTTST ;SAVE STARTING ADDRESS OF NEXT TEST
3373                                ;FOR ESCAPE ON PARITY ERRORS
3374 013742 012737 000020 001264 20$:    MOV      #TIMOUT, CPUEXP ;MIGHT TIME OUT OVER UNIBUS
3375 013750 012737 014006 001110      MOV      #10$, $LPERR    ;SET LOOP ON ERROR POINTER TO 10$
3376 013756 013700 172354              MOV      KIPAR6, R0      ;PUT UNIBUS ADDRESS OF MAP REG IN R0
3377 013762 072027 177773              ASH      #-5, R0         ;RIGHT SHIFT R0 5 PLACES
3378 013766 042700 007400              BIC      #007400, R0     ;STRIP OFF EXTRANEIOUS BITS.
3379 013772 012720 021042              MOV      #021042, (R0)+ ;SET BOTTOM BIT IN EACH ALU
3380 013776 012710 000042              MOV      #42, (R0)      ;SET BOTTOM BIT IN EACH ALU
3381 014002 005037 120000              CLR      #120000        ;CLEAR TEST LOCATION VIA FAST BUS
3382 014006 000240              10$:    NOP                    ;THIS IS A SYNC POINT FOR SCOPING
3383 014010 012737 043207 140000      MOV      #43207, #140000 ;LOAD TEST LOCATION VIA UNIBUS
3384                                ;THIS LOAD SHOULD NOT BE RELOCATED
3385                                ;BY THE UNIBUS MAP, SINCE BITCS OF
3386                                ;MMR3 IS CLEAR.
3387 014016 013703 120000              MOV      #120000, R3     ;READ TEST LOCATION VIA FAST BUS
3388 014022 022703 043207              CMP      #43207, R3     ;SEE IF DATA MATCHES
3389 014026 001401              BEQ      1$              ;BRANCH IF DATA GOOD
3390 014030 104032              ERROR   32               ;MAP RELOCATED WHEN NOT ENABLED
3391 014032 012737 013742 001110 1$:    MOV      #20$, $LPERR     ;SET LOOP POINTER TO START OF TEST
3392 014040 005037 001264              CLR      CPUEXP          ;ZERO EXPECTED CPU TRAP CONDITION
3393

```

# F06

3394  
3395  
3396  
3397  
3398  
3399  
3400  
3401  
3402  
3403  
3404  
3405  
3406  
3407  
3408  
3409  
3410  
3411  
3412  
3413  
3414  
3415  
3416  
3417  
3418  
3419  
3420  
3421  
3422  
3423  
3424  
3425  
3426  
3427  
3428  
3429  
3430  
3431  
3432  
3433  
3434  
3435  
3436  
3437  
3438  
3439  
3440  
3441  
3442  
3443  
3444  
3445  
3446  
3447  
3448  
3449

014044  
014044 000004  
014046 012737 014500 001324  
014054 012737 000001 001204  
014062  
014062 012737 014112 001106  
014070 012737 014112 001110  
014076 012737 000024 001102  
014104 013737 001102 177570  
014112 012737 000020 001264  
014120 012700 170200  
014124 012720 020000  
014130 005020  
014132 022700 170374  
014136 101372  
014140 052737 000040 172516  
014146 012700 117776  
014152 012737 170000 172350  
014160 012701 000200  
014164 012702 125252  
014170 005037 037776  
014174 010210  
014176 023702 037776  
014202 001411  
014204 060137 172350  
014210 022737 177600 172350  
014216 001364  
014220 104033  
014222 000137 010000  
014226 013737 172350 001240  
014234 013737 172350 001242  
014242 060137 172350  
014246 022737 177600 172350  
014254 001433  
014256 005037 037776  
014262 010210  
014264 023702 037776  
014270 001761  
014272 005037 037776  
014276 005237 001254  
014302 010210

```

*****
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
SIZEJ:  MOV      #1,$TIMES ;DO 1 ITERATION
MOV      #20,$SLPADR ;SET LOOP ON TEST POINTER TO 20$
MOV      #20,$SLPERR ;SET LOOP ON ERROR POINTER TO 20$
MOV      #24,$STSTNM ;SETUP TEST NUMBER AND CLR ERROR FLAG
MOV      $STSTNM,DISPLAY ;DISPLAY TEST NUMBER FOR ALL TO SEE
20$:  MOV      #TIMOUT,CPUEXP ;EXPECTING CPU TIME OUT ON UNIBUS
MOV      #MAPLO,R0 ;LOAD ADDRESS OF FIRST MAP REG
1$:  MOV      #20000,(R0)+ ;LOAD 4K INTO ALL MAP REGISTERS
CLR      (R0)+ ;INSURE THAT ALL REGS HAVE UPPER BITS CLR
CMP      #MAPL37,R0 ;SEE IF LAST REG IS LOADED
BHI      1$ ;BRANCH IF THERE ARE MORE TO LOAD
BIS      #BITS,$MMR3 ;TURN ON MAP RELOCATION
MOV      #117776,R0 ;THIS WILL BE USED TO SELECT PAR 4
MOV      #170000,KIPAR4 ;WE START TESTING WITH MAP 0
MOV      #200,R1 ;CONSTANT USED TO ADD TO PAR 4
MOV      #125252,R2 ;CONSTANT TO LOAD INTO LOCATION 37776
2$:  CLR      @#37776 ;CLEAR TEST LOCATION
MOV      R2,(R0) ;TRY TO LOAD TEST CELL THROUGH MAP
CMP      @#37776,R2 ;SEE IF TEST LOCATION WAS LOADED
BEQ      3$ ;BRANCH IF IT WAS LOADED
ADD      R1,KIPAR4 ;CELL NOT LOADED, TEST NEXT MAP REG
CMP      #177600,KIPAR4 ;SEE IF YOU'RE POINTING TO I/O PAGE
BNE      2$ ;GO TYPE NEXT MAP REGISTER
ERROR    33 ;FATAL ERROR RESTARTING PROGRAM
JMP      START ;RESTART PROGRAM
3$:  MOV      KIPAR4,LOWEST ;FOUND THE LOWEST USABLE MAP REG
4$:  MOV      KIPAR4,HIGEST ;THIS WILL END UP BEING LAST USABLE REG
ADD      R1,KIPAR4 ;TRY NEXT MAP REG TO SEE IF IT RESPONDS
CMP      #177600,KIPAR4 ;SEE IF ALL MAP REGS HAVE BEEN TRIED
BEQ      7$ ;BRANCH IF ALL ARE DONE
CLR      @#37776 ;CLEAR TEST LOCATION
MOV      R2,(R0) ;TRY TO LOAD TEST CELL THROUGH THE MAP
CMP      @#37776,R2 ;SEE IF TEST LOCATION WAS LOADED
BEQ      4$ ;BRANCH IF IT WAS LOADED
5$:  CLR      @#37776 ;CLEAR TEST LOCATION
INC      ERRCNT ;COUNT OF REGS AFTER LAST ONE USABLE
MOV      R2,(R0) ;TRY TO LOAD TEST CELL THRU MAP
  
```



3450	014304	023702	037776			CMP	#37776,R2	;SEE IF TEST LOCATION WAS LOADED
3451	014310	001402				BEQ	6\$	;BRANCH IF LOCATION WAS LOADED
3452	014312	005237	001256			INC	CNTR	;COUNT OF REGS THAT FAILED TO RESPOND
3453	014316	060137	172350		6\$:	ADD	R1,KIPAR4	;POINT TO NEXT MAP REG UNDER TEST
3454	014322	022737	177600	172350		CMP	#177600,KIPAR4	;SEE IF TEST IS OVER
3455	014330	001360				BNE	5\$	;BRANCH IF TEST NOT DONE
3456	014332	023737	001254	001256		CMP	ERRCNT,CNTR	;SEE IF TRIES = FAILURES
3457	014340	001401				BEQ	7\$	;BRANCH IF EQUAL
3458	014342	104034				ERROR	34	;MAP JUMPER COMPARE CIRCUIT BAD
3459	014344	005037	001254		7\$:	CLR	ERRCNT	;CLEAR TRIES COUNTER
3460	014350	005037	001256			CLR	CNTR	;CLEAR FAILURE COUNTER
3461	014354	005037	001264			CLR	CPUEXP	;NO CPU TRAPS EXPECTED IN NEXT TEST
3462	014350	005737	001100			TST	\$PASS	;SEE IF THIS IS FIRST PASS
3463	014364	001045				BNE	TST25	;GO TO NEXT TEST IF NOT THE FIRST PASS
3464	014366	023727	001240	170000		CMP	LOWEST,#170000	;SEE IF LOWER JUMPER IS DEFAULT
3465	014374	001004				BNE	8\$	;BRANCH IF NOT.
3466	014376	023727	001242	177400		CMP	HIGEST,#177400	;SEE IF UPPER JUMPER IS DEFAULT.
3467	014404	001401				BEQ	9\$	;BRANCH IF JUMPERS DEFAULT.
3468	014406	104035			8\$:	ERROR	35	;MAP SIZE JUMPERS NOT DEFAULT
3469								
3470								
3471								
3472								
3473	014410	013700	001242		9\$:	MOV	HIGEST,R0	
3474	014414	013701	001240			MOV	LOWEST,R1	
3475	014420	042700	170000			BIC	#170000,R0	
3476	014424	042701	170000			BIC	#170000,R1	
3477	014430	072027	177773			ASH	#-5,R0	;RIGHT SHIFT R0 5 PLACES
3478	014434	072127	177773			ASH	#-5,R1	;RIGHT SHIFT R1 5 PLACES
3479	014440	062701	170200			ADD	#170200,R1	
3480	014444	062700	170200			ADD	#170200,R0	
3481	014450	010137	001244			MOV	R1,LREGL	
3482	014454	062701	000002			ADD	#2,R1	;POINT TO UPPER BITS OF MAP REG
3483	014460	010137	001246			MOV	R1,LREGU	
3484	014464	010037	001250			MOV	R0,HREGL	
3485	014470	062700	000002			ADD	#2,R0	;POINT TO UPPER BITS OF MAP REG
3486	014474	010037	001252			MOV	R0,HREGU	
3487								
3488								
3489								
3490								
3491								
3492								
3493								
3494								
3495								
3496								
3497								
3498								
3499								
3500								
3501								
3502								
3503								
3504	014500							
3505	014500	000004						

```

*****
*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.
*****
*ST25: SCOPE

```

```

3506 014502 012737 014674 001324      MOV      #TST26,NXTTST ;SAVE STARTING ADDRESS OF NEXT TEST
3507                                     ;FOR ESCAPE ON PARITY ERRORS
3508 014510 012737 000144 001204      MOV      #144,$TIMES ;DO 144 ITERATIONS
3509 014516 004737 005054                20$: JSR      PC,CLRMAP ;CLEAR ALL MAP REGISTERS
3510 014522 005037 001170                CLR      $TMP0 ;USED AS FLAG IN TEST
3511 014526 012703 100000                MOV      #100000,R3 ;SELECT P.A.R. 4 OFFSET OF ZERO
3512 014532 012737 014564 001110      MOV      #2,$SLPERR ;SET LOOP ON ERROR POINTER TO 2$
3513 014540 013702 001244                MOV      LREG1,R2 ;PUT ADDRESS OF LOWEST USABLE MAP REG IN R2
3514 014544 013700 001240                MOV      LOWEST,R0 ;MAP REGISTER UNDER TEST IN R0
3515 014550 013701 001240                1$: MOV      LOWEST,R1 ;MAP REGISTER USED IN CURRENT REFERENCE
3516 014554 012712 040000                MOV      #40000,(R2) ;LOAD MAP REG UNDER TEST WITH 8K BASE
3517 014560 010237 040000                MOV      R2,#40000 ;LOAD TEST LOCATION WITH THE ADDRESS
3518                                     ;OF THE MAP REGISTER UNDER TEST
3519 014564 010137 172350                2$: MOV      R1,KIPAR4 ;LOAD PAR 4 WITH NEXT MAP REG UNIBUS ADDR
3520 014570 011304 (R3),R4 ;READ THROUGH THE MAP
3521 014572 020402 CMP      R4,R2 ;SEE IF CORRECT DATA WAS FETCHED
3522 014574 001010 BNE      4$ ;BRANCH IF NO MATCH
3523 014576 020001 CMP      R0,R1 ;SEE IF MAP REGS ARE THE SAME
3524 014600 001403 BEQ      3$ ;BRANCH IF CORRECT MAP REG WAS USED
3525 014602 004737 005276                JSR      PC,DUALADR ;LOG AND REPORT ALL ERRORS
3526 014606 000403 BR       4$ ;SKIP NEXT INSTRUCTION
3527 014610 012737 000001 001170    3$: MOV      #1,$TMP0 ;SET FLAG WHEN ADDRS MATCH
3528 014616 062701 000200                4$: ADD      #200,R1 ;TRY NEXT MAP REG
3529 014622 023701 001242                CMP      HIGEST,R1 ;SEE IF ALL HAVE BEEN TRIED
3530 014626 103356 BHS     2$ ;BRANCH IF STILL MORE TO TRY
3531 014630 005737 001170                TST      $TMP0 ;SEE THAT THERE WAS A SUCCESSFUL MATCH
3532 014634 001001 BNE      5$ ;BRANCH IF THERE WAS
3533 014636 104036 ERROR    36 ;DID NOT MATCH MAP REG ADDRESS
3534 014640 005037 001170                5$: CLR      $TMP0 ;CLEAR FLAG FOR NEXT REG
3535 014644 005012 CLR      (R2) ;CLEAR MAP REG JUST TESTED
3536 014646 062702 000004                ADD      #4,R2 ;POINT TO NEXT MAP REG TO LOAD
3537 014652 062700 000200                ADD      #200,R0 ;POINT TO NEXT MAP REG UNDER TEST
3538 014656 023700 001242                CMP      HIGEST,R0 ;SEE IF ALL MAP REGS HAVE BEEN TESTED
3539 014662 103332 BHS     1$ ;BRANCH IF STILL MORE TO TEST
3540 014664 005737 001254                TST      ERRCNT ;SEE IF THERE WERE ANY ERRORS
3541 014670 001401 BEQ      TST26 ;BRANCH TO NEXT TEST IF NO ERRORS
3542 014672 104013 ERROR    13 ;ERROR TYPE OUT ITEM 13

```

```

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

```

```

3555 TST26:
3556 014674 000004 SCOPE
3557 014674 012737 015154 001324      MOV      #TST27,NXTTST ;SAVE STARTING ADDRESS OF NEXT TEST
3558 014676 012737 015154 001324      MOV      #TST27,NXTTST ;FOR ESCAPE ON PARITY ERRORS
3559                                     ;TURN OFF MAP RELOCATION
3560 014704 042737 000040 172516      BIC      #BITS,MMR3 ;MAP PAGE 4 TO 8K
3561 014712 012737 000400 172350      MOV      #400,KIPAR4

```

```

3562 014720 012700 040000      MOV      #40000,R0      ;STARTING ADDRESS FOR DATA PATTERN
3563 014724 012701 100000      1$: MOV      #100000,R1    ;VIRTUAL ADDRESS
3564 014730 012702 010000      MOV      #104096,R2    ;LOAD 4096 LOCATIONS AT A TIME
3565 014734 010021 2$: MOV      R0,(R1)+    ;LOAD PHY. ADDR. INTO EACH MEMORY LOC.
3566 014736 062700 000002      ADD      #2,R0        ;POINT TO NEXT PHYSICAL ADDRESS
3567 014742 077204      SOB      R2,2$        ;BRANCH IF 4K OF MEMORY NOT LOADED
3568 014744 062737 000200 172350  ADD      #200,KIPAR4   ;POINT TO NEXT 4K BANK OF MEMORY
3569 014752 022737 001400 172350  CMP      #1400,KIPAR4  ;SEE IF 24K IS LOADED
3570 014760 101361      BHI      1$          ;BRANCH IF MORE MEMORY TO LOAD
3571      ;*
3572      ;*
3573      ;*
3574 014762 012737 014770 001106      MOV      #20$,SLPADR   ;SET LOOP ADDRESS TO 20$
3575 014770 012737 015026 001110 20$: MOV      #4$,SLPERR   ;SET LOOP ON ERROR POINTER TO 4$
3576 014776 022737 171400 172354  CMP      #171400,KIPAR6 ;DID I USE ANY MAP REGISTER
3577      ;BELOW REGISTER 6 (UB. ADDR 140000)
3578 015004 101463      BLOS    TST27        ;BRANCH TO NEXT TEST IF NOT
3579 015006 013700 172354      MOV      KIPAR6,R0    ;LOAD PAR6 INTO R0 TO GET
3580      ;THE STARTING DATA PATTERN
3581 015012 072027 000006      ASH      #6,R0        ;R0 NOW HOLDS THE STARTING DATA PATTERN
3582 015016 012701 140000      3$: MOV      #140000,R1  ;STARTING VIRTUAL ADDRESS
3583 015022 012702 010000      MOV      #104096,R2   ;PREPARE TO READ 4K AT A TIME
3584 015026 000240      4$: NOP              ;THIS IS A SYNC POINT FOR SCOPING
3585 015030 011103      MOV      (R1),R3      ;READ MAIN MEMORY THRU UNIBUS
3586 015032 020003      CMP      R0,R3        ;SEE IF THE ADDRESSES MATCH
3587 015034 001015      BNE      6$          ;BRANCH IF ERROR
3588 015036 062701 000002      5$: ADD      #2,R1      ;CHANGE VIRTUAL ADDRESS
3589 015042 062700 000002      ADD      #2,R0        ;CHANGE PHYSICAL ADDRESS
3590 015046 077211      SOB      R2,4$        ;BRANCH IF 4K OF MEMORY NOT READ
3591 015050 062737 000200 172354  ADD      #200,KIPAR6  ;POINT TO NEXT BANK OF 4K THRU UNIBUS
3592 015056 022737 171400 172354  CMP      #171400,KIPAR6 ;SEE IF THIS POINTS TO 24K PLUS 2
3593 015064 101354      BHI      3$          ;BRANCH IF 24K OF MEMORY NOT CHECKED
3594 015066 000423      BR       10$         ;TEST FINISHED, BRANCH TO EXIT
3595
3596 015070      6$:
3597 015070 005100      COM      R0           ;GET R0 READY FOR AND
3598 015072 040037 001224      BIC      R0,ADRAND    ;PERFORM LOGICAL AND
3599 015076 005100      COM      R0           ;PUT R0 BACK AS IT WAS
3600 015100 005103      COM      R3           ;GET R3 READY FOR AND
3601 015102 040337 001230      BIC      R3,DATAND    ;PERFORM LOGICAL AND
3602 015106 005103      COM      R3           ;PUT R3 BACK AS IT WAS
3603 015110 050037 001226      BIS      R0,ADDROR    ;LOGICAL OR OF PHYSICAL ADDRESS
3604 015114 050337 001232      BIS      R3,DATAOR    ;LOGICAL OR OF ADDRESS FETCHED
3605 015120 005737 001254      TST      ERRCNT      ;IS THIS THE FIRST ERROR HERE?
3606 015124 001002      BNE      7$          ;BRANCH IF NOT FIRST ERROR
3607 015126 104206      ERROR   206          ;DIDN'T READ ADDRESSES RIGHT FROM UNIBUS
3608 015130 000742      BR       5$          ;CONTINUE TESTING
3609 015132 104306      7$: ERROR   306          ;REPORT MORE DATA FROM UNIBUS
3610 015134 000740      BR       5$          ;CONTINUE WITH TEST
3611 015136 005737 001254      10$: TST      ERRCNT    ;WERE THERE ANY ERRORS ON THIS TEST?
3612 015142 001401      BEQ     19$          ;BRANCH IF NO ERRORS ON THIS TEST
3613 015144 104037      ERROR   37           ;SUMMARY OF UNIBUS ADDRESS FAILURES
3614 015146 012737 014770 001110 19$: MOV      #20$,SLPERR  ;SET LOOP POINTER TO 20$
3615
3616
3617
; ;*****

```

```

3618 : *TEST 27 MAIN MEMORY TIMEOUT THROUGH MAP
3619 : *
3620 : * THIS TEST GENERATES A TIME OUT THROUGH THE UNIBUS MAP BY TRYING
3621 : * . TO REFERENCE ADDRESS 17000000 IN MAIN MEMORY. IT USES THE LOWEST
3622 : * USEABLE MAP REGISTER, WHICH IN THE DEFAULT CASE IS MAP REGISTER
3623 : * ZERO.
3624 : *
3625 : *

```

```

3626 : *****
3627 TST27: SCOPE
3628 015154 000004 MOV #TST30,NXTTST ;SAVE STARTING ADDRESS OF NEXT TEST
3629 015156 012737 015266 001324 ;FOR ESCAPE ON PARITY ERRORS
3630 015164 052737 000040 172516 20$: BIS #BITS,MMR3 ;TURN MAP RELOCATION BACK ON
3631 015172 012737 000020 001264 MOV #TIMOUT,CPUEXP ;EXPECTING TIMEOUT IN THIS TEST
3632 015200 013737 001240 172350 MOV LOWEST,KIPAR4 ;LOAD PAR 4 WITH LOWEST USABLE MAP REG
3633 015206 012777 000074 164032 MOV #74,ALREGU ;LOAD UPPER 6 BITS OF LOWEST MAP REG
3634 015214 012777 000000 164022 MOV #000000,ALREGL ;LOAD LOWER 16 BITS OF LOWEST MAP REG
3635 015222 012737 015230 001110 MOV #1$,SLPERR ;SET LOOP ON ERROR POINTER TO 1$
3636 015230 005037 001266 1$: CLR PCPUER ;CPU ERROR REG LOCATION
3637 015234 000240 NOP ;THIS IS A SYNC POINT FOR SCOPING
3638 015236 013703 100000 MOV #100000,R3 ;TRY TO READ THRU PAGE 4
3639 ;THIS REFERENCE WILL GO OUT ON THE
3640 ;UNIBUS TO SELECT THE LOWEST USEABLE
3641 ;MAP REGISTER (DEFAULT MAP REG. 0).
3642 ;PHYSICAL ADDRESS 17700000 IS THEN
3643 ;GENERATED, WHICH SHOULD TIME OUT SINCE
3644 ;IT IS THE FIRST NON-EXISTANT LOCATION.
3645 015242 022737 000020 001266 CMP #TIMOUT,PCPUER ;THE UNIBUS SHOULD HAVE TIMED OUT
3646 015250 001401 BEQ 10$ ;BRANCH IF CONDITION WAS CORRECT
3647 015252 104040 ERROR 40 ;UNIBUS DID NOT TIME OUT
3648 015254 012737 015164 001110 10$: MOV #20$,SLPERR ;SET LOOP POINTER TO START OF TEST
3649 015262 005037 001264 CLR CPUEXP ;NO CPU TRAPS EXPECTED FOR AWHILE
3650
3651
3652 : *****

```

```

3653 : *TEST 30 RELOCATION TEST USING LOWEST USABLE MAPPING REG
3654 : *
3655 : * THIS TEST CHECKS OUT THE FULL ADDITION PROPERTIES OF THE UNIBUS
3656 : * MAP A.L.U.. IN THE DEFAULT CASE IT USES MAP REGISTER ZERO BUT
3657 : * IF THE MAP JUMPERS HAVE BEEN ALTERED TO DE-SELECT SOME MAP REGISTERS
3658 : * THIS TEST WILL USE THE LOWEST USEABLE MAP REGISTER.
3659 : * IF AN ERROR OCCURS THE TEST WILL REPORT THE PHYSICAL ADDRESS
3660 : * THAT WAS DESIRED, AND THE DATA AT THE ADDRESS THAT WAS REFERENCED.
3661 : *
3662 : *

```

```

3663 : *****
3664 TST30: SCOPE
3665 015266 000004 MOV #TST31,NXTTST ;SAVE STARTING ADDRESS OF NEXT TEST
3666 015270 012737 016174 001324 ;FOR ESCAPE ON PARITY ERRORS
3667
3668 : *THE RELOCATION HERE USES A BASE OF 00060000 AND AN
3669 : *OFFSET OF 00000 TO PRODUCE AN ADDRESS OF 00060000
3670
3671 100$: MOV #1$,SLPERR ;SET LOOP ON ERROR POINTER TO 1$
3672 015304 005077 163736 CLR ALREGU ;CLEAR UPPER BITS OF MAPPING REG
3673 015310 012777 060000 163726 MOV #060000,ALREGL ;LOAD LOWER BITS OF MAPPING REG

```

# K06

PDP-11.70 UNIBUS MAP DIAGNOSTIC MACY11 27(732) 20-SEP-76 13:32 PAGE 75  
 DEKBFB.P11 T30 RELOCATION TEST USING LOWEST USABLE MAPPING REG

```

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

```

# L06

```

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

M06

```

3796 015774 104041          ERROR 41          ;FAULTY RELOCATION BY UNIBUS MAP
3787
3798
3799
3790
3791 015776 012737 016030 001110 16$:  MOV    #17$, $LPERR      ;SET LOOP ON ERROR POINTER TO 17$
3792 016004 005077 163236          CLR    $LREGU          ;CLEAR UPPER BITS OF MAPPING REG
3793 016010 012777 057776 163226          MOV    #057776, $LREGL ;LOAD LOWER BITS OF MAPPING REG
3794 016016 013737 001240 172354          MOV    LOWEST, $#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
3795 016024 012700 141042          MOV    #141042, R0     ;SELECT PAR6, OFFSET IS 01042
3796 016030 000240          17$:  NOP                    ;THIS IS A SYNC POINT FOR SCOPING
3797 016032 011001          MOV    (R0), R1        ;READ LOCATION 061040 THRU THE UNIBUS
3798 016034 012702 061040          MOV    #061040, R2    ;THE EXPECTED PHYSICAL ADDRESS IS 061040
3799 016040 020102          CMP    R1, R2         ;SEE IF THE MAP'S FETCH WAS CORRECT
3800 016042 001401          BEQ   18$             ;BRANCH IF FETCHED DATA MATCHES ADDRESS
3801 016044 104041          ERROR 41             ;FAULTY RELOCATION BY UNIBUS MAP
3802
3803
3804
3805
3806 016046 012737 016100 001110 18$:  MOV    #19$, $LPERR      ;SET LOOP ON ERROR POINTER TO 19$
3807 016054 005077 163166          CLR    $LREGU          ;CLEAR UPPER BITS OF MAPPING REG
3808 016060 012777 041042 163156          MOV    #041042, $LREGL ;LOAD LOWER BITS OF MAPPING REG
3809 016066 013737 001240 172354          MOV    LOWEST, $#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
3810 016074 012700 157776          MOV    #157776, R0     ;SELECT PAR6, OFFSET IS 17776
3811 016100 000240          19$:  NOP                    ;THIS IS A SYNC POINT FOR SCOPING
3812 016102 011001          MOV    (R0), R1        ;READ LOCATION 061040 THRU THE UNIBUS
3813 016104 012702 061040          MOV    #061040, R2    ;THE EXPECTED PHYSICAL ADDRESS IS 061040
3814 016110 020102          CMP    R1, R2         ;SEE IF THE MAP'S FETCH WAS CORRECT
3815 016112 001401          BEQ   20$             ;BRANCH IF FETCHED DATA MATCHES ADDRESS
3816 016114 104041          ERROR 41             ;FAULTY RELOCATION BY UNIBUS MAP
3817
3818
3819
3820
3821 016116 012737 016150 001110 20$:  MOV    #21$, $LPERR      ;SET LOOP ON ERROR POINTER TO 21$
3822 016124 005077 163116          CLR    $LREGU          ;CLEAR UPPER BITS OF MAPPING REG
3823 016130 012777 057776 163106          MOV    #057776, $LREGL ;LOAD LOWER BITS OF MAPPING REG
3824 016136 013737 001240 172354          MOV    LOWEST, $#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
3825 016144 012700 140002          MOV    #140002, R0     ;SELECT PAR6, OFFSET IS 00002
3826 016150 000240          21$:  NOP                    ;THIS IS A SYNC POINT FOR SCOPING
3827 016152 011001          MOV    (R0), R1        ;READ LOCATION 060000 THRU THE UNIBUS
3828 016154 012702 060000          MOV    #060000, R2    ;THE EXPECTED PHYSICAL ADDRESS IS 060000
3829 016160 020102          CMP    R1, R2         ;SEE IF THE MAP'S FETCH WAS CORRECT
3830 016162 001401          BEQ   22$             ;BRANCH IF FETCHED DATA MATCHES ADDRESS
3831 016164 104041          ERROR 41             ;FAULTY RELOCATION BY UNIBUS MAP
3832 016166 012737 015276 001110 22$:  MOV    #100$, $LPERR    ;SET LOOP POINTER TO START OF TEST
3833
3834
3835
3836
3837
3838
3839
3840
3841

```

;\*\*\*\*\*  
;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



```

3842          ;*      FOR THE TOP OF MAIN MEMORY.
3843          ;*
3844          ;*****
3845 016174    TST31:
3846 016174 000004          SCOPE
3847 016176 012737 016464 001324      MOV      #TST32,NXTTST      ;SAVE STARTING ADDRESS OF NEXT TEST
3848          ;FOR ESCAPE ON PARITY ERRORS
3849 016204 012737 000012 001204      MOV      #12,$TIMES        ;DO 12 ITERATIONS
3850          ;BIT05 IS NON-EXISTANT MEMORY BIT IN
3851          ;THE CPU ERROR REGISTER
3852 016212 012737 177777 016346 20$:  MOV      #-1,4$          ;INITIALIZE FLAG AS NEGATIVE ONE
3853 016220 005077 163022          CLR      @LREGU           ;CLEAR UPPER 6 BITS OF MAP REG
3854 016224 012777 020000 163012      MOV      #20000,@LREGL    ;LOAD 4K BASE INTO MAP REGISTER
3855 016232 012701 100100          MOV      #100100,R1       ;LOAD BITS TO SELECT PAR 4, OFFSET 100
3856 016236 012700 150000          MOV      #150000,R0       ;LOAD BITS TO SELECT PAR 6, OFFSET 2K
3857 016242 012737 000277 172350      MOV      #277,KIPAR4     ;START WITH PHYSICAL 6K
3858 016250 013737 001240 172354      MOV      LOWEST,KIPAR6    ;LOAD PAR 6 WITH MAP REG'S ADDR
3859 016256 012737 016316 001110      MOV      #10$, $LPERR     ;SET LOOP ON ERROR POINTER TO 10$
3860 016264 012737 000020 001264 3$:  MOV      #20,$CPUEXP      ;EXPECTING A UNIBUS TIME OUT DURING TEST
3861 016272 005037 001266          CLR      PCPUER          ;CLEAR TIME OUT FLAG
3862 016276 013710 001326          MOV      DATA,(R0)      ;THIS LOAD WILL TIME OUT WHEN YOU
3863          ;HAVE REACHED THE TOP OF MEMORY
3864          ;IT SELECTS PAR 6 WHICH WILL PUT ADDR
3865          ;<XX XX>0000 ON THE UNIBUS.
3866          ;THE X'S WILL SELECT THE LOWEST USEABLE
3867          ;MAPPING REGISTER. THE DEFAULT CASE IS
3868          ;010000, SELECTING MAPPING REGISTER 0.
3869 016302 005737 001266          TST      PCPUER          ;SEE IF THERE WAS MAIN MEMORY
3870 016306 001016          BNE     1$              ;BRANCH IF NO MAIN MEMORY FROM UNIBUS
3871 016310 012737 000040 001264      MOV      #NEXMEM,$CPUEXP ;POSSIBLE CACHE NON-EXISTANT MEMORY
3872 016316 011103          MOV      (R1),R3         ;READ TEST LOCATION VIA FASTBUS
3873 016320 022737 000040 001266 10$:  CMP      #NEXMEM,PCPUER  ;WAS THIS CACHE NON-EXISTANT MEMORY
3874 016326 001414          BEQ     2$              ;BRANCH IF NON-EXISTANT MEMORY
3875 016330 000240          NOP                     ;THIS IS A SYNC POINT FOR SCOPING
3876 016332 011002          MOV      (R0),R2        ;READ TEST LOCATION VIA UNIBUS MAP
3877 016334 020203          CMP      R2,R3         ;COMPARE TEST DATA R2=MAP DATA
3878          ;R3=FASTBUS DATA
3879 016336 001410          BEQ     2$              ;BRANCH IF IT WAS THE SAME
3880 016340 104042          ERROR  42              ;CARRY PROPAGATION FAILURE IN MAP
3881 016342 000406          BR      2$              ;BRANCH TO UPDATE ROUTINE
3882 016344 005227          1$:  INC      (PC)+          ;INCREMENT ONE TIME ENTRANCE FLAG
3883 016346 177777          4$:  .WORD  -1              ;USE NEGATIVE ONE FOR FLAG
3884 016350 001003          BNE     2$              ;BRANCH IF YOU'VE BEEN HERE BEFORE
3885 016352 013737 172350 001320      MOV      KIPAR4,$RSIZE   ;SAVE UPPER LIMIT OF MEMORY
3886 016360 062737 000100 001326 2$:  ADD      #100,DATA        ;CHANGE PATTERN FOR NEXT LOAD
3887 016366 062737 000100 172350      ADD      #100,KIPAR4     ;ADD 2K TO PAR4
3888 016374 062777 010000 162642      ADD      #10000,@LREGL   ;ADD 2K TO MAP REGISTER
3889 016402 001330          BNE     3$              ;BRANCH IF MAP REGISTER NOT ZERO
3890 016404 005277          INC      @LREGU         ;ADD ONE TO UPPER 6 BITS OF MAP REG
3891 016410 022777 000073 162630      CMP      #73,@LREGU     ;SEE IF TOP 128K BLOCK HAS BEEN PASSED
3892 016416 103322          BHIS   3$              ;BRANCH IF NOT PAST IT
3893 016420 005237 001326          INC      DATA          ;CHANGE DATA PATTERN FOR NEXT PASS
3894 016424 042737 177700 001326      BIC      #177700,DATA    ;CLEAR UPPER 10 BITS OF DATA PATTERN
3895 016432 052737 000300 001326      BIS      #300,DATA       ;START WITH 3XX IN DATA PATTERN
3896 016440 012737 016212 001110      MOV      #20$, $LPERR    ;SET LOOP POINTER TO START OF TEST
3897 016446 023737 177760 001320      CMP      @#SIZELO,$RSIZE ;SEE IF SIZE JUMPERS AGREE WITH MEMORY TOP

```





3954	016674	013737	177742	001272
3955	016702	013737	177744	001274
3956	016710	013737	177746	001276
3958	016716	012737	030000	001300
3959	016724	000002		

```

MOV 2#HIACRS,PHIADR ;SAVE HIGH BITS OF FAILING ADDR
MOV 2#MEMERR,PPARER ;SAVE MEMORY ERROR REGISTER
MOV 2#CONTRL,PCONTR ;SAVE CONTROL REGISTER FOR TYPE OUT
MOV 2#30000,PMINT ;SAVE DATA IN MAINTENANCE REGISTER
RTI ;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
VECTOR WILL TRAP TO LABEL 10$ AT THE END OF THIS TEST.

```

```

*****
TST33:

```

3975	016726			
3976	016726	000004		
3977	016730	012737	017200	001324
3978				
3979	016736	012737	023410	001200
3980	016744	005037	001274	
3981	016750	012737	177777	177744
3982	016756	013737	177746	001202
3983	016764	012737	000014	177746
3984	016772	013737	001240	172354
3985	017000	012777	040000	162236
3986	017006	005077	162234	
3987	017012	012701	177750	
3988	017016	012737	017054	001110
3989	017024	005037	140000	
3990	017030	012700	140002	
3991	017034	012737	017054	001110
3992	017042	012710	020001	
3993	017046	012737	017140	000114
3994	017054	012737	140000	177750
3995	017062	000240		
3996	017064	011003		
3997	017066	011102		
3998	017070	005011		
3999	017072	050000		
4000	017074	023737	001200	001274
4001	017102	001401		
4002	017104	104044		
4003	017106	012737	177777	177744
4004	017114	012737	005704	000114
4005	017122	013737	001202	177746
4006	017130	012737	016736	001110
4007	017136	000420		
4008				
4009				

```

SCOPE
MOV 2#TST34,NXTTST ;SAVE STARTING ADDRESS OF NEXT TEST
;FOR ESCAPE ON PARITY ERRORS
20$: MOV 2#23410,$TMP4 ;LOAD EXPECTED ERROR CONDITION
CLR PPARER ;CLEAR MEM ERROR REG STORAGE
MOV 2#-1,$MEMERR ;CLEAR MEMORY ERROR REGISTER
MOV 2#CONTRL,$TMP5 ;SAVE CONTROL REG TO RESTORE CACHE
MOV 2#14,$CONTRL ;FORCE MISSES IN CACHE
MOV LOWEST,KIPAR6 ;PUT UNIBUS ADDR OF MAP REG IN PAR6
MOV 2#40000,$LREGL ;LOWEST MAP REGISTER POINTS TO 8K
CLR 2#LREGU ;CLEAR UPPER BITS OF MAP REG
MOV 2#MAINT,R1 ;PUT ADDR OF MAINT REG IN R1
MOV 2#15,$LPERR ;SET LOOP ON ERROR POINTER TO 15
CLR 140000 ;CLEAR LOCATION 40000
MOV 2#140002,R0 ;LOAD ADDRESS OF 40002 INTO R0
MOV 2#15,$LPERR ;SET LOOP ON ERROR POINTER TO 15
MOV 2#20001,(R0) ;LOAD ONE BIT IN EACH BYTE OF 40002
MOV 2#105,CACHVEC ;SET PARITY VECTOR TO 105
15: MOV 2#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 ;DUMMY INST WITH P.B.'S ON
CMP $TMP4,PPARER ;SEE IF MEM ERROR REG HOLDS RIGHT DATA
BEQ 25 ;BRANCH IF ERROR WAS EXPECTED ONE
ERROR 44 ;PARITY REPORTING BAD
25: MOV 2#-1,MEMERR ;CLEAR MEMORY ERROR REGISTER
MOV 2#MEMER,CACHVEC ;RESTORE PARITY SERVICE ROUTINE
MOV $TMP5,CONTRL ;RESTORE CACHE TO FORMER CONDITION
MOV 2#205,$LPERR ;SET LOOP POINTER TO START OF TEST
BR TST34 ;TEST IS OVER, GO TO NEXT TEST

```

```

4010 017140 013737 177740 001270 10$: MOV      2#LOADRS,PLOADR ;SAVE LOWER CACHE ADDR REG
4011 017146 013737 177742 001272      MOV      2#HIADRS,PHIADR ;SAVE HIGH BITS OF FAILING ADDR
4012 017154 013737 177744 001274      MOV      2#MEMERR,PPARER ;SAVE MEMORY ERROR REGISTER
4013 017162 013737 177746 001276      MOV      2#CONTRL,PCONTR ;SAVE CONTROL REGISTER FOR TYPE OUT
4014 017170 012737 140000 001300      MOV      #140000,PMaint ;SAVE DATA IN MAINTENANCE REGISTER
4015 017176 000002                      RTI      ;RETURN TO TEST AND CHECK PARITY

```

4016  
4017  
4018  
4019  
4020  
4021  
4022  
4023  
4024  
4025  
4026  
4027  
4028  
4029  
4030  
4031  
4032

```

*****
*TEST 34      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.

```

```

4033 017200
4034 017200 000004
4035 017202 012737 017466 001324
4036
4037 017210 012737 177777 017354 20$: MOV      #-1,21$
4038 017216 032737 000004 177746      BIT      #BIT2,2#CONTRL ;SEE IF GROUP 0 IS DISABLED
4039 017224 001120                      BNE     TST35 ;BRANCH IF GROUP 0 IS DISABLED
4040 017226 013737 177746 001202      MOV      CONTRL,$TMP5 ;SAVE CONDITION OF CACHE
4041 017234 012737 023504 001200      MOV      #23504,$TMP4 ;EXPECTED PARITY CONDITION GROUP 0
4042 017242 005037 001274                      CLR     PPARER ;CLEAR MEM ERROR REG STORAGE
4043 017246 012737 177777 177744      MOV      #-1,2#MEMERR ;CLEAR MEMORY ERROR REGISTER
4044 017254 013737 001240 172354      MOV      LOWEST,KIPAR6 ;PUT UNIBUS ADDR OF MAP REG IN PAR6
4045 017262 005077 161760                      CLR     2LREGU ;CLEAR UPPER BITS ON MAP REG
4046 017266 012777 040000 161750      MOV      #40000,2LREGL ;LOAD LOWER 16 BITS OF MAP REG
4047 017274 012705 177750                      MOV     #MAINT,R5 ;PUT ADDR OF MAINT REG IN R5
4048 017300 012704 140000                      MOV     #140000,R4 ;SELECT PAR 6 BASE 0
4049 017304 012714 100000                      MOV     #BIT15,(R4) ;MAKE SURE I GET SOFT AND HARD ERRORS
4050
4051
4052

```

```

*****
TST34:
SCOPE
MOV      #TST35,NXTTST ;SAVE STARTING ADDRESS OF NEXT TEST
;FOR ESCAPE ON PARITY ERRORS
MOV      #-1,21$ ;RESTORE NEG ONE FOR NEXT ITERATION
BIT      #BIT2,2#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      #-1,2#MEMERR ;CLEAR MEMORY ERROR REGISTER
MOV      LOWEST,KIPAR6 ;PUT UNIBUS ADDR OF MAP REG IN PAR6
CLR     2LREGU ;CLEAR UPPER BITS ON MAP REG
MOV      #40000,2LREGL ;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     #30,2#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$,CACHVEC ;SET PARITY VECTOR TO 10$
15: MOV     R1,2#MAINT ;FORCE GPO PARITY BITS TO 0 ON 2ND PASS
CLC ;THIS IS A SYNC POINT FOR SCOPING
;THE LOW BYTE HAS ITS P.B. OFF
MOV     (R4),R1 ;DATA FETCH SHOULD CAUSE PARITY ABORT
MOV     (R5),R3 ;SAVE MAINT REG IN CASE NO TRAP
CLRB   (R5) ;CLEAR MAINT REG IN CASE NO TRAP
SXT    R1 ;DUMMY INST WITH P.B.'S OFF
MOV     #020020,R1 ;FORCE GROUP 0 LOW BYTE PARITY BIT
;TO 0 AND MAIN MEMORY EVEN WORD HIGH

```

```

4053 017310 012737 000030 177746      MOV     #30,2#CONTRL
4054 017316 011401                      MOV     (R4),R1
4055 017320 005001                      CLR     R1
4056 017322 012737 017426 000114      MOV     #10$,CACHVEC
4057 017330 010137 177750 15: MOV     R1,2#MAINT
4058 017334 000241                      CLC
4059
4060 017336 011401                      MOV     (R4),R1
4061 017340 011503                      MOV     (R5),R3
4062 017342 105015                      CLRB   (R5)
4063 017344 006701                      SXT    R1
4064 017346 012701 020020      MOV     #020020,R1
4065

```

```

4066
4067 017352 005227
4068 017354 177777
4069 017356 001764
4070 017360 023737 001200 001274
4071 017366 001402
4072 017370 010302
4073 017372 104044
4074 017374 012737 177777 177744
4075 017402 012737 005704 000114
4076 017410 013737 001202 177746
4077 017416 012737 017210 001110
4078 017424 000420
4079
4080
4081 017426 013737 177740 001270
4082 017434 013737 177742 001272
4083 017442 013737 177744 001274
4084 017450 013737 177746 001276
4085 017456 012737 030060 001300
4086 017464 000002
4087
4088
4089
4090
4091
4092
4093
4094
4095
4096
4097
4098
4099
4100
4101
4102
4103
4104 017466
4105 017466 000004
4106 017470 012737 020032 001324
4107
4108 017476 012737 177777 017642
4109 017504 032737 000010 177746
4110 017512 001120
4111 017514 013737 177746 001202
4112 017522 012737 023604 001200
4113 017530 005037 001274
4114 017534 012737 177777 177744
4115 017542 013737 001240 172354
4116 017550 005077 161472
4117 017554 012777 040000 161462
4118 017562 012705 177750
4119 017566 012704 140000
4120 017572 012714 100000
4121

```

```

                INC      (PC)+
21$:          .WORD    -1
                BEQ     1$
                CMP     $TMP4,PPARER
                BEQ     2$
                MOV     R3,R2
                ERROR   44
2$:          MOV     #-1, MEMERR
                MOV     #MEMER,CACHVEC
                MOV     $TMP5,CONTRL
                MOV     #20$, $LPERR
                BR      TST35
;BYTE PARITY BIT TO 1
;INCREMENT NEXT WORD TO ZERO ON FIRST PASS
;BRANCH ON FIRST PASS ONLY
;SEE IF ERROR CONDITION MATCHES EXPECTED
;BRANCH IF CORRECT CONDITION
;R2 HOLDS MAINT REG FOR ERROR TYPE OUT
;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

10$:         MOV     @#LCADRS,PLOADR ;SAVE LOWER CACHE ADDR REG
                MOV     @#HIADRS,PHIADR ;SAVE HIGH BITS OF FAILING ADDR
                MOV     @#MEMERR,PPARER ;SAVE MEMORY ERROR REGISTER
                MOV     @#CONTRL,PCONTR ;SAVE CONTROL REGISTER FOR TYPE OUT
                MOV     #30060,PMaint ;SAVE DATA IN 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:
                SCOPE
                MOV     #TST36,NXTTST ;SAVE STARTING ADDRESS OF NEXT TEST
;FOR ESCAPE ON PARITY ERRORS
20$:         MOV     #-1,21$ ;RESTORE NEG ONE FOR NEXT ITERATION
                BIT     #BIT3,CONTRL ;SEE IF GROUP 1 IS DISABLED
                BNE    UBMAP ;BRANCH TO NEXT TEST IF GROUP 1 DISABLED
                MOV     CONTRL,$TMP5 ;SAVE CONDITION OF CACHE
                MOV     #23604,$TMP4 ;EXPECTED PARITY CONDITION GROUP 1
                CLR     PPARER ;CLEAR MEM ERROR REG STORAGE
                MOV     #-1,@#MEMERR ;CLEAR MEMORY ERROR REGISTER
                MOV     LOWEST,KIPAR6 ;PUT UNIBUS ADDR OF MAP REG IN PAR6
                CLR     @LREGU ;CLEAR UPPER BITS ON MAP REG
                MOV     #40000,@LREGL ;LOAD LOWER 16 BITS OF MAP REG
                MOV     #MAINT,R5 ;PUT ADDR OF MAINT REG IN R5
                MOV     #140000,R4 ;SELECT PAR 6 BASE 0
                MOV     #BIT15,(R4) ;MAKE SURE I GET SOFT AND HARD ERRORS
;BY LOADING 40000 WITH BIT 15, THIS

```

# F07

```

4122                                     ;CAUSES THE HIGH BYTE P.B. TO BE ON
4123                                     ;AND THE LOW BYTE TO BE OFF
4124 017576 012737 000044 177746      MOV    #44,2#CONTRL ;FORCE SELECTION OF GROUP 0 ON READ
4125 017604 011401                    MOV    (R4),R1      ;PUT (40000) & (40002) IN GROUP 1
4126 017606 005001                    CLR    R1           ;CLEAR R1 FOR MOMENT
4127 017610 012737 017714 000114      MOV    #10$,CACHVEC ;SET PARITY VECTOR TO 10$
4128 017616 010137 177750 15:        MOV    R1,2#MAINT  ;FORCE GPI PARITY BITS TO 0 ON 2ND PASS
4129 017622 000241                    CLC                ;THIS IS A SYNC POINT FOR SCOPING
4130                                     ;THE LOW BYTE HAS ITS P.B. OFF
4131 017624 011401                    MOV    (R4),R1      ;DATA FETCH SHOULD CAUSE PARITY ABORT
4132 017626 011503                    MOV    (R5),R3      ;SAVE MAINT REG IN CASE NO TRAP
4133 017630 105015                    CLR    (R5)         ;CLEAR MAINT REG IN CASE NO TRAP
4134 017632 006701                    SXT    R1           ;DUMMY INST WITH P.B.'S OFF
4135 017634 012701 020100            MOV    #020100,R1   ;FORCE GROUP 1 LOW BYTE PARITY BIT
4136                                     ;TO 0 AND MAIN MEMORY EVEN WORD HIGH
4137                                     ;BYTE PARITY BIT TO 1
4138 017640 005227                    INC    (PC)+        ;INCREMENT NEXT WORD TO 0
4139 017642 177777 215:              .WORD  -1
4140 017644 001764                    BEQ    15           ;BRANCH ON FIRST PASS ONLY
4141 017646 023737 001200 001274      CMP    $TMP4,PPARER ;SEE IF ERROR CONDITION MATCHES EXPECTED
4142 017654 001402                    BEQ    25           ;BRANCH IF ERROR WAS EXPECTED ONE
4143 017656 010302                    MOV    R3,R2        ;R2 HOLDS MAINT REG FOR ERROR TYPE OUT
4144 017660 104044                    ERROR  44           ;PARITY REPORTING BAD
4145 017662 012737 177777 177744 25: MOV    #-1,MEMERR    ;CLEAR MEMORY ERROR REGISTER
4146 017670 012737 005704 000114      MOV    #MEMER,CACHVEC ;RESTORE PARITY SERVICE ROUTINE
4147 017676 013737 001202 177746      MOV    $TMP5,CONTRL ;RESTORE CACHE TO FORMER CONDITION
4148 017704 012737 017476 001110      MOV    #20$,LPERR   ;SET LOOP POINTER TO START OF TEST
4149 017712 000420                    BR     UBMAP        ;;TEST OVER GO TO NEXT TEST
4150
4151
4152 017714 013737 177740 001270 105:  MOV    2#LOADRS,PLOADR ;SAVE LOWER CACHE ADDR REG
4153 017722 013737 177742 001272      MOV    2#HIADRS,PHIADR ;SAVE HIGH BITS OF FAILING ADDR
4154 017730 013737 177744 001274      MOV    2#MEMERR,PPARER ;SAVE MEMORY ERROR REGISTER
4155 017736 013737 177746 001276      MOV    2#CONTRL,PCONTR ;SAVE CONTROL REGISTER FOR TYPE OUT
4156 017744 012737 030300 001300      MOV    #30300,PMAINT ;SAVE DATA IN MAINTENANCE REGISTER
4157 017752 000002                    RTI                ;RETURN TO TEST AND CHECK PARITY
4158
4159
4160
4161 .SBTTL *****
4162 .SBTTL THE FOLLOWING TESTS ARE RUN THROUGH THE UNIBUS MAP
4163 .SBTTL *****
4164
4165
4166 017754 000004      UBMAP: SCOPE          ;LOOP ON PREVIOUS TEST
4167 017756 104420   TBITR             ;RESTORE T-BIT IF IT WAS ON
4168
4169
4170
4171
4172
4173
4174
4175
4176
4177

```

```

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

```

```

4178          :*
4179 017760 013701 001246          MOV  LREGU,R1          ;PUT POINTER TO LOWEST MAP REG IN R1
4180 017764 005061 000004          CLR  4(R1)           ;CLEAR UPPER 6 BITS OF (LOWEST + 1)
4181 017770 005061 000010          CLR  10(R1)          ;CLEAR UPPER 6 BITS OF (LOWEST + 2)
4182 017774 012761 020000 000002  MOV  #20000,2(R1)     ;LOAD LOWER 16 BITS OF (LOWEST + 1)
4183                                     ;SO THAT IT POINTS TO 4K - 9K
4184 020002 005061 000006          CLR  6(R1)           ;CLEAR LOWER 16 BITS OF (LOWEST + 2)
4185 020006 013701 001240          MOV  LOWEST,R1       ;PREPARE TO LOAD PAR1
4186 020012 062701 000200          ADD  #200,R1         ;POINTER TO (LOWEST + 1)
4187 020016 010137 172342          MOV  R1,KIPAR1       ;LOAD PAR1 TO POINT TO LOWEST + 1
4188 020022 062701 000200          ADD  #200,R1         ;ADJUST R1 FOR KIPAR0
4189 020026 010137 172340          MOV  R1,KIPAR0       ;LOAD PAR0 TO POINT TO (LOWEST + 2)
4190
4191
4192          ;*****
4193          ;*TEST 36          MAIN MEMORY TIMEOUT THRU MAP, CODE RUN OVER UNIBUS
4194          ;*
4195          ;*          THIS TEST GENERATES A TIME OUT THROUGH THE UNIBUS MAP BY TRYING
4196          ;*          TO REFERENCE ADDRESS 17000000 IN MAIN MEMORY. IT USES THE LOWEST
4197          ;*          USEABLE MAP REGISTER, WHICH IN THE DEFAULT CASE IS MAP REGISTER
4198          ;*          ZERO.
4199          ;*
4200          ;*          THIS TEST IS BEING RUN WITH ALL MEMORY REFERENCES GOING THRU
4201          ;*          THE UNIBUS MAP.
4202          ;*
4203          ;*****
4204          ;*ST36:
4205 020032 012737 020070 001106          MOV  #20$, $LPADR     ;SET LOOP ON TEST POINTER TO 20$
4206 020040 012737 020070 001110          MOV  #20$, $LPERR     ;SET LOOP ON ERROR POINTER TO 20$
4207 020046 012737 000036 001102          MOV  #36,$STSTNM      ;SETUP TEST NUMBER AND CLR ERROR FLAG
4208 020054 013737 001102 177570          MOV  $STSTNM,DISPLAY  ;DISPLAY TEST NUMBER FOR ALL TO SEE
4209 020062 012737 020164 001324          MOV  #TST37,NXTTST    ;SET UP ESCAPE VECTOR IN CASE OF PARITY ERRORS
4210 020070 012737 000020 001264 20$:  MOV  #20,CPUEXP        ;EXPECTING CPU TIMEOUT IN THIS TEST
4211 020076 013737 001240 172350          MOV  LOWEST,KIPAR4    ;LOAD PAR 4 WITH LOWEST USABLE MAP REG
4212 020104 012777 000074 161134          MOV  #74,$LREGU      ;LOAD UPPER 6 BITS OF LOWEST MAP REG
4213 020112 012777 000000 161124          MOV  #000000,$LREGL  ;LOAD LOWER 16 BITS OF LOWEST MAP REG
4214 020120 012737 020126 001110          MOV  #1$, $LPERR     ;SET LOOP ON ERROR POINTER TO 1$
4215 020126 005037 001266 1$:          CLR  PCPUER          ;CPU ERROR REG LOCATION
4216 020132 000240          NOP                   ;THIS IS A SYNC POINT FOR SCOPING
4217 020134 013703 100000          MOV  #100000,R3      ;TRY TO READ THRU PAGE 4
4218                                     ;THIS REFERENCE WILL GO OUT ON THE
4219                                     ;UNIBUS TO SELECT THE LOWEST USEABLE
4220                                     ;MAP REGISTER (DEFAULT MAP REG. 0).
4221                                     ;PHYSICAL ADDRESS 17700000 IS THEN
4222                                     ;GENERATED, WHICH SHOULD TIME OUT SINCE
4223                                     ;IT IS THE FIRST NON-EXISTANT LOCATION.
4224 020140 022737 000020 001266          CMP  #20,PCPUER       ;THE UNIBUS SHOULD HAVE TIMED OUT
4225 020146 001401          BEQ  10$             ;BRANCH IF UNIBUS TIMED OUT
4226 020150 104045          ERROR 45             ;DID NOT TIME OUT OVER UNIBUS
4227 020152 012737 020070 001110 10$:  MOV  #20$, $LPERR     ;SET LOOP POINTER TO START OF TEST
4228 020160 005037 001264          CLR  CPUEXP          ;NO CPU TRAPS EXPECTED IN NEXT TEST
4229
4230
4231          ;*****
4232          ;*TEST 37          RELOCATION TEST USING LOWEST USABLE MAPPING REG
4233          ;*

```



H07

4234 : \* THIS TEST CHECKS OUT THE FULL ADDITION PROPERTIES OF THE UNIBUS  
4235 : \* MAP A.L.U.. IN THE DEFAULT CASE IT USES MAP REGISTER ZERO BUT  
4236 : \* IF THE MAP JUMPERS HAVE BEEN ALTERED TO DE-SELECT SOME MAP REGISTERS  
4237 : \* THIS TEST WILL USE THE LOWEST USEABLE MAP REGISTER.  
4238 : \* IF AN ERROR OCCURS THE TEST WILL REPORT THE PHYSICAL ADDRESS  
4239 : \* THAT WAS DESIRED, AND THE DATA AT THE ADDRESS THAT WAS REFERENCED.

4240 : \* THIS TEST IS BEING RUN WITH ALL MEMORY REFERENCES GOING THRU  
4241 : \* THE UNIBUS MAP.  
4242 : \*  
4243 : \*  
4244 : \*  
4245 : \*\*\*\*\*

```
4245 020164  
4246 020164 000004  
4247 020166 012737 021100 001324  
4248  
4249 020174 012737 060000 060000  
4250  
4251  
4252  
4253  
4254  
4255 020202 012737 020234 001110  
4256 020210 005077 161032  
4257 020214 012777 060000 161022  
4258 020222 013737 001240 172354  
4259 020230 012700 140000  
4260 020234 000240  
4261 020236 011001  
4262 020240 012702 060000  
4263 020244 020102  
4264 020246 001401  
4265 020250 104046  
4266  
4267  
4268  
4269  
4270 020252 012737 020304 001110  
4271 020260 005077 160762  
4272 020264 012777 052524 160752  
4273 020272 013737 001240 172354  
4274 020300 012700 145252  
4275 020304 000240  
4276 020306 011001  
4277 020310 012702 057776  
4278 020314 020102  
4279 020316 001401  
4280 020320 104046  
4281  
4282  
4283  
4284  
4285 020322 012737 020354 001110  
4286 020330 005077 160712  
4287 020334 012777 045252 160702  
4288 020342 013737 001240 172354  
4289 020350 012700 152524
```

```
TST37: SCOPE  
MOV #TST40,NXTTST ;SAVE STARTING ADDRESS OF NEXT TEST  
;FOR ESCAPE ON PARITY ERRORS  
MOV #060000,#060000 ;MAKE SURE THAT ADDRESS 060000  
;CONTAINS ITS OWN ADDRESS AS DATA
```

THE RELOCATION HERE USES A BASE OF 00060000 AND AN  
OFFSET OF 00000 TO PRODUCE AN ADDRESS OF 00060000

```
100$: MOV #15,$LPERR ;SET LOOP ON ERROR POINTER TO 15  
CLR $LR$GU ;CLEAR UPPER BITS OF MAPPING REG  
MOV #060000,$LR$GL ;LOAD LOWER BITS OF MAPPING REG  
MOV LOWEST,$KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG  
MOV #140000,R0 ;SELECT PAR6, OFFSET IS 00000  
1$: NOP ;THIS IS A SYNC POINT FOR SCOPING  
MOV (R0),R1 ;READ LOCATION 060000 THRU THE UNIBUS  
MOV #060000,R2 ;THE EXPECTED PHYSICAL ADDRESS IS 060000  
CMP R1,R2 ;SEE IF THE MAP'S FETCH WAS CORRECT  
BEQ 2$ ;BRANCH IF FETCHED DATA MATCHES ADDRESS  
ERROR 46 ;FAULTY RELOCATION BY UNIBUS MAP
```

THE RELOCATION HERE USES A BASE OF 00052524 AND AN  
OFFSET OF 05252 TO PRODUCE AN ADDRESS OF 00057776

```
2$: MOV #35,$LPERR ;SET LOOP ON ERROR POINTER TO 35  
CLR $LR$GU ;CLEAR UPPER BITS OF MAPPING REG  
MOV #052524,$LR$GL ;LOAD LOWER BITS OF MAPPING REG  
MOV LOWEST,$KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG  
MOV #145252,R0 ;SELECT PAR6, OFFSET IS 05252  
3$: NOP ;THIS IS A SYNC POINT FOR SCOPING  
MOV (R0),R1 ;READ LOCATION 057776 THRU THE UNIBUS  
MOV #057776,R2 ;THE EXPECTED PHYSICAL ADDRESS IS 057776  
CMP R1,R2 ;SEE IF THE MAP'S FETCH WAS CORRECT  
BEQ 4$ ;BRANCH IF FETCHED DATA MATCHES ADDRESS  
ERROR 46 ;FAULTY RELOCATION BY UNIBUS MAP
```

THE RELOCATION HERE USES A BASE OF 00045252 AND AN  
OFFSET OF 12524 TO PRODUCE AN ADDRESS OF 00057776

```
4$: MOV #55,$LPERR ;SET LOOP ON ERROR POINTER TO 55  
CLR $LR$GU ;CLEAR UPPER BITS OF MAPPING REG  
MOV #045252,$LR$GL ;LOAD LOWER BITS OF MAPPING REG  
MOV LOWEST,$KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG  
MOV #152524,R0 ;SELECT PAR6, OFFSET IS 12524
```

```

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

```



4346	020570	005077	160452		CLR	2LREGU	; CLEAR UPPER BITS OF MAPPING REG
4347	020574	012777	056734	160442	MOV	#056734, 2LREGL	; LOAD LOWER BITS OF MAPPING REG
4348	020602	013737	001240	172354	MOV	LOWEST, 2#KIPAR6	; LOAD PAR6 WITH ADDR OF LOWEST MAP REG
4349	020610	012700	142104		MOV	#142104, R0	; SELECT PAR6, OFFSET IS 02104
4350	020614	000240			13\$: NOP		; THIS IS A SYNC POINT FOR SCOPING
4351	020616	011001			MOV	(R0), R1	; READ LOCATION 061040 THRU THE UNIBUS
4352	020620	012702	061040		MOV	#061040, R2	; THE EXPECTED PHYSICAL ADDRESS IS 061040
4353	020624	020102			CMP	R1, R2	; SEE IF THE MAP'S FETCH WAS CORRECT
4354	020626	001401			BEQ	14\$	; BRANCH IF FETCHED DATA MATCHES ADDRESS
4355	020630	104046			ERROR	46	; FAULTY RELOCATION BY UNIBUS MAP

::  
 ;\*THE RELOCATION HERE USES A BASE OF 00042104 AND AN  
 ;\*OFFSET OF 16734 TO PRODUCE AN ADDRESS OF 00061040

4360	020632	012737	020664	001110	14\$: MOV	#15\$, \$LPERR	; SET LOOP ON ERROR POINTER TO 15\$
4361	020640	005077	160402		CLR	2LREGU	; CLEAR UPPER BITS OF MAPPING REG
4362	020644	012777	042104	160372	MOV	#042104, 2LREGL	; LOAD LOWER BITS OF MAPPING REG
4363	020652	013737	001240	172354	MOV	LOWEST, 2#KIPAR6	; LOAD PAR6 WITH ADDR OF LOWEST MAP REG
4364	020660	012700	156734		MOV	#156734, R0	; SELECT PAR6, OFFSET IS 16734
4365	020664	000240			15\$: NOP		; THIS IS A SYNC POINT FOR SCOPING
4366	020666	011001			MOV	(R0), R1	; READ LOCATION 061040 THRU THE UNIBUS
4367	020670	012702	061040		MOV	#061040, R2	; THE EXPECTED PHYSICAL ADDRESS IS 061040
4368	020674	020102			CMP	R1, R2	; SEE IF THE MAP'S FETCH WAS CORRECT
4369	020676	001401			BEQ	16\$	; BRANCH IF FETCHED DATA MATCHES ADDRESS
4370	020700	104046			ERROR	46	; FAULTY RELOCATION BY UNIBUS MAP

::  
 ;\*THE RELOCATION HERE USES A BASE OF 00057776 AND AN  
 ;\*OFFSET OF 01042 TO PRODUCE AN ADDRESS OF 00061040

4375	020702	012737	020734	001110	16\$: MOV	#17\$, \$LPERR	; SET LOOP ON ERROR POINTER TO 17\$
4376	020710	005077	160332		CLR	2LREGU	; CLEAR UPPER BITS OF MAPPING REG
4377	020714	012777	057776	160322	MOV	#057776, 2LREGL	; LOAD LOWER BITS OF MAPPING REG
4378	020722	013737	001240	172354	MOV	LOWEST, 2#KIPAR6	; LOAD PAR6 WITH ADDR OF LOWEST MAP REG
4379	020730	012700	141042		MOV	#141042, R0	; SELECT PAR6, OFFSET IS 01042
4380	020734	000240			17\$: NOP		; THIS IS A SYNC POINT FOR SCOPING
4381	020736	011001			MOV	(R0), R1	; READ LOCATION 061040 THRU THE UNIBUS
4382	020740	012702	061040		MOV	#061040, R2	; THE EXPECTED PHYSICAL ADDRESS IS 061040
4383	020744	020102			CMP	R1, R2	; SEE IF THE MAP'S FETCH WAS CORRECT
4384	020746	001401			BEQ	18\$	; BRANCH IF FETCHED DATA MATCHES ADDRESS
4385	020750	104046			ERROR	46	; FAULTY RELOCATION BY UNIBUS MAP

::  
 ;\*THE RELOCATION HERE USES A BASE OF 00041042 AND AN  
 ;\*OFFSET OF 17776 TO PRODUCE AN ADDRESS OF 00061040

4390	020752	012737	021004	001110	18\$: MOV	#19\$, \$LPERR	; SET LOOP ON ERROR POINTER TO 19\$
4391	020760	005077	160262		CLR	2LREGU	; CLEAR UPPER BITS OF MAPPING REG
4392	020764	012777	041042	160252	MOV	#041042, 2LREGL	; LOAD LOWER BITS OF MAPPING REG
4393	020772	013737	001240	172354	MOV	LOWEST, 2#KIPAR6	; LOAD PAR6 WITH ADDR OF LOWEST MAP REG
4394	021000	012700	157776		MOV	#157776, R0	; SELECT PAR6, OFFSET IS 17776
4395	021004	000240			19\$: NOP		; THIS IS A SYNC POINT FOR SCOPING
4396	021006	011001			MOV	(R0), R1	; READ LOCATION 061040 THRU THE UNIBUS
4397	021010	012702	061040		MOV	#061040, R2	; THE EXPECTED PHYSICAL ADDRESS IS 061040
4398	021014	020102			CMP	R1, R2	; SEE IF THE MAP'S FETCH WAS CORRECT
4399	021016	001401			BEQ	20\$	; BRANCH IF FETCHED DATA MATCHES ADDRESS
4400	021020	104046			ERROR	46	; FAULTY RELOCATION BY UNIBUS MAP

::

# K07

```

4402 ;*THE RELOCATION HERE USES A BASE OF 00057776 AND AN
4403 ;*OFFSET OF 00002 TO PRODUCE AN ADDRESS OF 00060000
4404
4405 021022 012737 021054 001110 20$: MOV #21$, $LPERR ;SET LOOP ON ERROR POINTER TO 21$
4406 021030 005077 160212 CLR $LREGU ;CLEAR UPPER BITS OF MAPPING REG
4407 021034 012777 057776 160202 MOV #057776, $LREGL ;LOAD LOWER BITS OF MAPPING REG
4408 021042 013737 001240 172354 MOV LOWEST, $KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
4409 021050 012700 140002 MOV #140002, R0 ;SELECT PAR6, OFFSET IS 00002
4410 021054 000240 21$: NOP ;THIS IS A SYNC POINT FOR SCOPING
4411 021056 011001 MOV (R0), R1 ;READ LOCATION 060000 THRU THE UNIBUS
4412 021060 012702 060000 MOV #060000, R2 ;THE EXPECTED PHYSICAL ADDRESS IS 060000
4413 021064 020102 CMP R1, R2 ;SEE IF THE MAP'S FETCH WAS CORRECT
4414 021066 001401 BEQ 22$ ;BRANCH IF FETCHED DATA MATCHES ADDRESS
4415 021070 104046 ERROR 46 ;FAULTY RELOCATION BY UNIBUS MAP
4416 021072 012737 020202 001110 22$: MOV #100$, $LPERR ;SET LOOP POINTER TO START OF TEST
4417
4418
4419
4420
4421
4422
4423
4424
4425
4426
4427
4428
4429
4430
4431
4432
4433
4434
4435
4436
4437
4438
4439
4440
4441
4442
4443
4444
4445
4446
4447
4448
4449
4450
4451
4452
4453
4454
4455
4456
4457

```

```

*****
;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:
*****

```

```

4431 021100
4432 021100 000004
4433 021102 012737 021370 001324
4434
4435 021110 012737 000012 001204
4436 021116 012737 177777 021252 20$: MOV #12, $TIMES ;DO 12 ITERATIONS
4437 021124 005077 160116 CLR $LREGU ;INITIALIZE FLAG AS NEGATIVE ONE
4438 021130 012777 020000 160106 MOV #20000, $LREGL ;CLEAR UPPER 6 BITS OF MAP REG
4439 021136 012701 100100 MOV #100100, R1 ;LOAD 4K BASE INTO MAP REGISTER
4440 021142 012700 150000 MOV #150000, R0 ;LOAD BITS TO SELECT PAR 4, OFFSET 100
4441 021146 012737 000277 172350 MOV #277, KIPAR4 ;LOAD BITS TO SELECT PAR 6, OFFSET 2K
4442 021154 013737 001240 172354 MOV LOWEST, KIPAR6 ;START WITH PHYSICAL 6K
4443 021162 012737 021222 001110 MOV #10$, $LPERR ;LOAD PAR 6 WITH MAP REG'S ADDR
4444 021170 012737 000020 001264 3$: MOV #TIMOUT, CPUEXP ;SET LOOP ON ERROR POINTER TO 10$
4445 021176 005037 001266 CLR PCPUER ;EXPECTING A UNIBUS TIME OUT DURING TEST
4446 021202 013710 001326 MOV DATA, (R0) ;CLEAR TIME OUT FLAG
4447 ;THIS LOAD WILL TIME OUT WHEN YOU
4448 ;HAVE REACHED THE TOP OF MEMORY
4449 ;IT SELECTS PAR 6 WHICH WILL PUT ADDR
4450 ;<XXX XX1>0000 ON THE UNIBUS.
4451 ;THE X'S WILL SELECT THE LOWEST USEABLE
4452 ;MAPPING REGISTER. THE DEFAULT CASE IS
4453 ;010000, SELECTING MAPPING REGISTER 0.
4454 021206 005737 001266 TST PCPUER ;SEE IF THERE WAS MAIN MEMORY
4455 021212 001016 BNE 1$ ;BRANCH IF NO MAIN MEMORY FROM UNIBUS
4456 021214 012737 000040 001264 10$: MOV #NEXMEM, CPUEXP ;POSSIBLE CACHE NON-EXISTANT MEMORY
4457 021222 011103 MOV (R1), R3 ;READ TEST LOCATION VIA FASTBUS
021224 022737 000040 001266 CMP #NEXMEM, PCPUER ;WAS THERE CACHE NON-EXISTANT MEMORY

```

4458	021232	001414				BEQ	2\$		; BRANCH IF NON-EXISTANT MEMORY
4459	021234	000240				NOP			; THIS IS A SYNC POINT FOR SCOPING
4460	021236	011002				MOV	(R0), R2		; READ TEST LOCATION VIA UNIBUS MAP
4461	021240	020203				CMP	R2, R3		; COMPARE TEST DATA R2=MAP DATA
4462									; R3=FASTBUS DATA
4463	021242	001410				BEQ	2\$		; BRANCH IF IT WAS THE SAME
4464	021244	104047				ERROR	47		; FAILURE IN CARRY PROPAGATION IN MAP
4465	021246	000406				BR	2\$		; BRANCH TO UPDATE ROUTINE
4466	021250	005227			1\$:	INC	(PC)+		; INCREMENT ONE TIME ENTRANCE FLAG
4467	021252	177777			4\$:	.WORD	-1		; USE NEGATIVE ONE FOR FLAG
4468	021254	001003				BNE	2\$		; BRANCH IF YOU'VE BEEN HERE BEFORE
4469	021256	013737	172350	001320		MOV	KIPAR4, RSIZE		; SAVE UPPER LIMIT OF MEMORY
4470	021264	062737	000100	001326	2\$:	ADD	#100, DATA		; CHANGE PATTERN FOR NEXT LOAD
4471	021272	062737	000100	172350		ADD	#100, KIPAR4		; ADD 2K TO PAR4
4472	021300	062777	010000	157736		ADD	#10000, ALREGL		; ADD 2K TO MAP REGISTER
4473	021306	001330				BNE	3\$		; BRANCH IF MAP REGISTER NOT ZERO
4474	021310	005277	157732			INC	ALREGU		; ADD ONE TO UPPER 6 BITS OF MAP REG
4475	021314	022777	000073	157724		CMP	#73, ALREGU		; SEE IF TOP 128K BLOCK HAS BEEN PASSED
4476	021322	103322				BHIS	3\$		; BRANCH IF NOT PAST IT
4477	021324	005237	001326			INC	DATA		; CHANGE DATA PATTERN FOR NEXT PASS
4478	021330	042737	177700	001326		BIC	#177700, DATA		; CLEAR UPPER 10 BITS OF DATA PATTERN
4479	021336	052737	000300	001326		BIS	#300, DATA		; START WITH 3XX IN DATA PATTERN
4480	021344	012737	021116	001110		MOV	#20\$, \$LPERR		; SET LOOP POINTER TO START OF TEST
4481	021352	023737	177760	001320		CMP	2#SIZELO, RSIZE		; SEE IF SIZE JUMPERS AGREE WITH MEMORY TOP
4482	021360	001401				BEQ	19\$		; BRANCH IF TOP OF MEMORY AGREES WITH
4483									; THE SIZE JUMPERS
4484	021362	104050				ERROR	50		; MEMORY SIZE JUMPERS NOT RIGHT
4485	021364	005037	001264		19\$:	CLR	CPUEXP		; NO CPU TRAPS EXPECTED IN NEXT TEST

```

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

```

4503	021370					SCOPE			
4504	021370	000004				MOV	#TST42, NXTTST		; SAVE STARTING ADDRESS OF NEXT TEST
4505	021372	012737	021632	001324					; FOR ESCAPE ON PARITY ERRORS
4506						TBIT0			; TURN OFF T BIT TRAPPING IF ON
4507	021400	104416			20\$:	MOV	#23404, \$TMP4		; EXPECTED ERROR CONDITION
4508	021402	012737	023404	001200		CLR	PPARER		; CLEAR MEM ERROR REG STORAGE
4509	021410	005037	001274			MOV	#-1, MEMERR		; CLEAR MEMORY ERROR REGISTER
4510	021414	012737	177777	177744		MOV	2#CONTRL, \$TMP5		; SAVE CONTROL REG TO RESTORE CACHE
4511	021422	013737	177746	001202		MOV	#14, 2#CONTRL		; FORCE MISSES IN CACHE
4512	021430	012737	000014	177746		MOV	LOWEST, KIPAR6		; PUT UNIBUS ADDR OF MAP REG IN PAR6
4513	021436	013737	001240	172354					

# M07

PDP-11/70 UNIBUS MAP DIAGNOSTIC MACY11 27(732) 20-SEP-76 13:32 PAGE 90  
 DEKBFB.P11 T41 PARITY REPORTING THRU THE MAP, MAIN MEMORY EVEN WORD

```

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

```

N07

```

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

```

4636	0222114	012737	177777	022260	208:	MOV	#-1,215	:FOR ESCAPE ON PARITY ERRORS
4637	0222122	032737	000004	177746		BIT	#BIT2,2#CONTRL	:RESTORE NEG ONE FOR NEXT ITERATION
4638	0222130	001120				BNE	TST44	:SEE IF GROUP 0 IS DISABLED
4639	0222132	013737	177746	001202		MOV	CONTRL,STMP5	:BRANCH IF GROUP 0 IS DISABLED
4640	0222140	012737	023504	001200		MOV	#23504,STMP4	:SAVE CONDITION OF CACHE
4641	0222146	005037	001274			CLR	PPARER	:EXPECTED PARITY CONDITION GROUP 0
4642	0222152	012737	177777	177744		MOV	#-1,2#MEMERR	:CLEAR MEM ERROR REG STORAGE
4643	0222160	013737	001240	172354		MOV	LOWEST,KIPAR6	:CLEAR MEMORY ERROR REGISTER
4644	0222166	005037	157054			CLR	2LREGU	:PLT UNIBUS ADDR OF MAP REG IN PAR6
4645	0222172	012737	040000	157044		MOV	#40000,2LREGL	:CLEAR UPPER BITS ON MAP REG
4646	0222200	012735	177750			MOV	#MAINT,R5	:LOAD LOWER 16 BITS OF MAP REG
4647	0222204	012704	140000			MOV	#140000,R4	:PUT ADDR OF MAINT REG IN R5
4648	0222210	012714	100000			MOV	#BIT15,R4	:SELECT PAR 6 BASE 0
4649								:MAKE SURE I GET SOFT AND HARD ERRORS
4650								:BY LOADING 40000 WITH BIT 15, THIS
4651								:CAUSES THE HIGH BYTE P.B. TO BE ON
4652								:AND THE LOW BYTE P.B. TO BE OFF
4653	0222214	012737	000030	177746		MOV	#30,2#CONTRL	:FORCE SELECTION OF GROUP 0 ON READ
4654	0222222	011401				MOV	(R4),R1	:PLT (40000) & (40002) IN GROUP 0
4655	0222224	005001				CLR	R1	:ZERO R1 FOR THE MOMENT
4656	0222226	012737	022332	000114		MOV	#108,CACHVEC	:SET PARITY VECTOR TO 108
4657	0222234	010137	177750		18:	MOV	R1,2#MAINT	:FORCE GPO PARITY BITS TO 0 ON 2ND PASS
4658	0222240	000241				CLC		:THIS IS A SYNC POINT FOR SCOPING
4659								:THE LOW BYTE HAS ITS P.B. OFF
4660	0222242	011401				MOV	(R4),R1	:DATA FETCH SHOULD CAUSE PARITY ABORT
4661	0222244	011503				MOV	(R5),R3	:SAVE MAINT REG IN CASE NO TRAP
4662	0222246	105015				CLRB	R5	:CLEAR MAINT REG IN CASE NO TRAP
4663	0222250	006701				SXT	R1	:DUMMY INST WITH P.B.'S OFF
4664	0222252	012701	020020			MOV	#020020,R1	:FORCE GROUP 0 LOW BYTE PARITY BIT
4665								:TO 0 AND MAIN MEMORY EVEN WORD HIGH
4666								:BYTE PARITY BIT TO 1
4667	0222256	005227				INC	(PC)+	:INCREMENT NEXT WORD TO ZERO ON FIRST PASS
4668	0222260	177777			218:	.WORD	-1	
4669	0222262	001764				BEG	18	:BRANCH ON FIRST PASS ONLY
4670	0222264	023737	001200	001274		CMP	STMP4,PPARER	:SEE IF ERROR CONDITION MATCHES EXPECTED
4671	0222272	001402				BEG	25	:BRANCH IF CORRECT CONDITION
4672	0222274	010302				MOV	R3,R2	:R2 HOLDS MAINT REG FOR ERROR TYPE OUT
4673	0222276	104051				ERROR	51	:FAILURE IN PARITY REPORTING
4674	0222300	012737	177777	177744	25:	MOV	#-1,MEMERR	:CLEAR MEMORY ERROR REGISTER
4675	0222306	012737	005704	000114		MOV	#MEMER,CACHVEC	:RESTORE PARITY SERVICE ROUTINE
4676	0222314	013737	001202	177746		MOV	STMP5,CONTRL	:RESTORE CACHE TO FORMER CONDITION
4677	0222322	012737	022114	001110		MOV	#208,\$LPERR	:SET LOOP POINTER TO START OF TEST
4678	0222330	000420				BR	TST44	:TEST OVER GO TO NEXT TEST
4679								
4680								
4681	0222332	013737	177740	001270	108:	MOV	2#LOADRS,PLOADR	:SAVE LOWER CACHE ADDR REG
4682	0222340	013737	177742	001272		MOV	2#HIADRS,PHIADR	:SAVE HIGH BITS OF FAILING ADDR
4683	0222346	013737	177744	001274		MOV	2#MEMERR,PPARER	:SAVE MEMORY ERROR REGISTER
4684	0222354	013737	177746	001276		MOV	2#CONTRL,PCONTR	:SAVE CONTROL REGISTER FOR TYPE OUT
4685	0222362	012737	030060	001300		MOV	#30060,PMaint	:SAVE DATA IN MAINTENANCE REGISTER
4686	0222370	000002				RTI		:RETURN TO TEST AND CHECK PARITY

\*\*\*\*\*  
 : TEST 44 PARITY REPORTING THRU THE MAP, CACHE DATA GROUP 1  
 : \*



```

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

```

```

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

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

THIS TEST IS BEING RUN WITH ALL MEMORY REFERENCES GOING THRU
THE UNIBUS MAP.
*****
T44: SCOPE
MOV #SEOP,NXTTST ;SET UP ESCAPE POINTER IN CASE OF
;RANDOM PARITY ABORTS
20$: MOV #-1,21$ ;RESTORE NEG ONE FOR NEXT ITERATION
BIT #BIT3,CONTR ;SEE IF GROUP 1 IS DISABLED
BNE SEOP ;BRANCH TO NEXT TEST IF GROUP 1 DISABLED
MOV CONTR,$TMP5 ;SAVE CONDITION OF CACHE
MOV #23604,$TMP4 ;EXPECTED PARITY CONDITION GROUP 1
CLR PPARER ;CLEAR MEM ERROR REG STORAGE
MOV #-1,$MEMERR ;CLEAR MEMORY ERROR REGISTER
MOV LOWEST,KIPAR6 ;PUT UNIBUS ADDR OF MAP REG IN PAR6
CLR $LREGU ;CLEAR UPPER BITS ON MAP REG
MOV #40000,$LREGL ;LOAD LOWER 16 BITS OF MAP REG
MOV $MAINT,R5 ;PUT ADDR OF MAINT REG IN R5
MOV #140000,R4 ;SELECT PAR 6 BASE 0
MOV #BIT15,(R4) ;MAKE SURE I GET SOFT AND HARD ERRORS
;BY LOADING 40000 WITH BIT 15. THIS
;CAUSES THE HIGH BYTE P.B. TO BE ON
;AND THE LOW BYTE P.B. TO BE OFF
MOV #44,$CONTR ;FORCE SELECTION OF GROUP 0 ON READ
MOV (R4),R1 ;PUT (40000) & (40002) IN GROUP 1
CLR R1 ;CLEAR R1 FOR MOMENT
MOV #10$,CACHVEC ;SET PARITY VECTOR TO 10$
1$: MOV R1,$MAINT ;FORCE GP1 PARITY BITS TO 0 ON 2ND PASS
CLC ;THIS IS A SYNC POINT FOR SCOPING
;THE LOW BYTE HAS ITS P.B. OFF
MOV (R4),R1 ;DATA FETCH SHOULD CAUSE PARITY ABORT
MOV (R5),R3 ;SAVE MAINT REG IN CASE NO TRAP
CLRB (R5) ;CLEAR MAINT REG IN CASE NO TRAP
SXT R1 ;DUMMY INST WITH P.B.'S OFF
MOV #020100,R1 ;FORCE GROUP 1 LOW BYTE PARITY BIT
;TO 0 AND MAIN MEMORY EVEN WORD HIGH
;BYTE PARITY BIT TO 1
INC (PC)+ ;INCREMENT NEXT WORD TO 0
21$: .WORD -1
BEQ 1$ ;BRANCH ON FIRST PASS ONLY
CMP $TMP4,PPARER ;SEE IF ERROR CONDITION MATCHES EXPECTED
BEQ 2$ ;BRANCH IF ERROR WAS EXPECTED ONE
MOV R3,R2 ;R2 HOLDS MAINT REG FOR ERROR TYPE OUT
4735: ERROR 51 ;FAILURE IN PARITY REPORTING
4737: MOV #-1,MEMERR ;CLEAR MEMORY ERROR REGISTER

```

```

4738 022574 012737 005704 000114      MOV      #MEMER,CACHVEC ;RESTORE PARITY SERVICE ROUTINE
4739 022602 013737 001202 177746      MOV      $TMP5,CNTRL   ;RESTORE CACHE TO FORMER CONDITION
4740 022610 012737 022402 001110      MOV      #20$, $LPERR ;SET LOOP POINTER TO START OF TEST
4741 022616 000420                BR       $EOP          ;;TEST OVER GO TO NEXT TEST
4742
4743
4744 022620 013737 177740 001270 10$:  MOV      @#LOADRS,PLOADR ;SAVE LOWER CACHE ADDR REG
4745 022626 013737 177742 001272      MOV      @#HIADRS,PHIADR ;SAVE HIGH BITS OF FAILING ADDR
4746 022634 013737 177744 001274      MOV      @#MEMERR,PPARER ;SAVE MEMORY ERROR REGISTER
4747 022642 013737 177746 001276      MOV      @#CNTRL,PCONTR ;SAVE CONTROL REGISTER FOR TYPE OUT
4748 022650 012737 030300 001300      MOV      #30300,PMaint ;SAVE DATA IN MAINTENANCE REGISTER
4749 022656 000002                RTI                   ;RETURN TO TEST AND CHECK PARITY
4750
4751
4752
4753                ;;*****
4754
4755                .SBTTL  END OF PASS ROUTINE
4756
4757                ;*INCREMENT THE PASS NUMBER ($PASS)
4758                ;*INDICATE END-OF-PROGRAM AFTER 1 PASSES THRU THE PROGRAM
4759                ;*TYPE "END PASS #XXXXX TOTAL NUMBER OF ERRORS SINCE LAST REPORT YYYYYY"
4760                ;*WHERE XXXXX AND YYYYYY ARE DECIMAL NUMBERS
4761                ;*IF SW12=1 INHIBIT TRACE TRAP
4762                ;*IF THERES A MONITOR GO TO IT
4763                ;*IF THERE ISN'T JUMP TO LOOP
4764
4765                $EOP:
4766 022660 000004                SCOPE                ;LOOP ON LAST TEST
4767 022662 005037 177572      CLR      MMRO          ;TURN OFF FULL RELOCATION
4768 022666 005037 172516      CLR      MMR3          ;DISABLE THE UNIBUS MAP
4769 022672 104420                TBITR                ;RESTORE THE T BIT IF IT WAS ON
4770 022674 005037 001102      CLR      $TSTNM        ;ZERO THE TEST NUMBER
4771 022700 005037 001204      CLR      $TIMES        ;ZERO THE NUMBER OF ITERATIONS
4772 022704 005237 001100      INC      $PASS         ;INCREMENT THE PASS NUMBER
4773 022710 042737 100000 001100      BIC      #100000,$PASS ;DON'T ALLOW A NEG. NUMBER
4774 022716 005327                DEC      (PC)+        ;LOOP?
4775 022720 000001      $EOPCT: .WORD 1
4776 022722 003072      BGT      $DOAGN        ;;YES
4777 022724 012737      MOV      (PC)+,@(PC)+ ;RESTORE COUNTER
4778 022726 000001      $ENDCT: .WORD 1
4779 022730 022720                $EOPCT
4780 022732 104400 022740      TYPE    ,65$          ;;TYPE ASCIZ STRING
4781 022736 000407      BR      64$           ;;GET OVER THE ASCIZ
4782                ;;65$: .ASCIZ <12><15>/END PASS #/
4783                64$:
4784 022756 013746 001100      MOV      $PASS,-(SP)  ;;SAVE $PASS FOR TYPEOUT
4785                ;;TYPE PASS NUMBER
4786 022762 104410                TYPDS                ;;GO TYPE--DECIMAL ASCII WITH SIGN
4787 022764 104400 022772      TYPE    ,67$          ;;TYPE ASCIZ STRING
4788 022770 000421      BR      66$           ;;GET OVER THE ASCIZ
4789                ;;67$: .ASCIZ / TOTAL ERRORS SINCE LAST REPORT /
4790                66$:
4791 023034 013746 001112      MOV      $ERTTL,-(SP) ;;SAVE $ERTTL FOR TYPEOUT
4792                ;;TOTAL NUMBER OF ERRORS
4793 023040 104410                TYPDS                ;;GO TYPE--DECIMAL ASCII WITH SIGN
    
```



4794	023042	104400	001215		TYPE	\$.SCLF	:: TYPE CARRIAGE RETURN, LINE FEED
4795	023046	005037	001112		CLR	\$.SERTTL	:: CLEAR ERROR TOTAL
4796	023052	013700	000042	\$GET42:	MOV	2#42,RO	:: GET MONITOR ADDRESS
4797	023056	001414			BEQ	\$.SDOAGN	:: BRANCH IF NO MONITOR
4798	023060	005046			CLR	-(SP)	:: INSURE THE "T" BIT IS CLEAR
4799	023062	012746	023070		MOV	\$.SCLR.T,-(SP)	:: SETUP FOR AN RTI OR RTT
4800	023066	000427			SR	\$.SRTN	:: GO DO AN RTI OR RTT TO LOAD THE PSW
4801							:: WITH A CLEARED "T" BIT
4802	023070			\$CLR.T:			
4803	023070	013700	000042		MOV	2#42,RO	:: INSURE RO CONTAINS THE MONITORS
4804	023074	001405			BEQ	\$.SDOAGN	:: RETURN ADDRESS
4805	023076	000005			RESET		:: CLEAR THE WORLD
4806	023100	004710		\$ENDAD:	JSR	PC,(RO)	:: GO TO MONITOR
4807	023102	000240			NOP		:: SAVE ROOM
4808	023104	000240			NOP		:: FOR
4809	023106	000240			NOP		:: ACT11
4810	023110			\$.SDOAGN:			
4811	023110	013746	177776		MOV	2#PS,-(SP)	:: PUT THE PS ON THE STACK AND
4812	023114	042716	000020		BIC	20,(SP)	:: CLEAR THE "T" BIT
4813	023120	032737	010000	177570	BIT	2#BIT12,2#SWR	:: RUN WITH TRACE TRAP?
4814	023126	001005			BNE	IS	:: BR IF NO
4815	023130	005137	023154		COM	\$.STBIT	:: IS IT TIME FOR TRACE TRAP
4816	023134	100402			BMI	IS	:: BR IF NO
4817	023136	052716	000020		BIS	20,(SP)	:: SET TRACE TRAP
4818	023142	012746	023150	IS:	MOV	\$.SLOOP,-(SP)	:: JUMP TO START OF TEST
4819	023146	000002		\$SRTN:	RTI		:: RETURN--THIS IS CHANGED TO
4820							:: AN "RTT" IF "RTT" IS A LEGAL
4821							:: INSTRUCTION
4822	023150			\$.SLOOP:			
4823	023150	000137	010346		JMP	2#LOOP	:: RETURN
4824	023154	000000		\$.STBIT:	.WORD	0	:: "T" BIT STATE INDICATOR
4825	023156	377	377	000	\$.SENULL:	.BYTE	-1,-1,0
4826		023162			.EVEN		:: NULL CHARACTER STRING
4827							
4828							
4829				\$.SBTTL	ERROR MESSAGES AND DATA TABLES		
4830	023162	047516	020124	044124	EM1:	.ASCIZ	?NOT THE CORRECT TRAP CONDITION THRU ERRVEC (#004)?
4831	023170	020105	047503	051122			
4832	023176	041505	020124	051124			
4833	023204	050101	041440	047117			
4834	023212	044504	044524	047117			
4835	023220	052040	051110	020125			
4836	023226	051105	053122	041505			
4837	023234	024040	030043	032060			
4838	023242	000051					
4839	023244	047125	054105	042520	EM2:	.ASCIZ	?UNEXPECTED CPU TRAP THRU ERRVEC (#004)?
4840	023252	052103	042105	041440			
4841	023260	052520	052040	040522			
4842	023266	020120	044124	052522			
4843	023274	042440	051122	042526			
4844	023302	020103	021450	030060			
4845	023310	024464	000				
4846	023313	125	042516	050130	EM3:	.ASCIZ	?UNEXPECTED CACHE PARITY ERROR THRU CACHVEC, WILL RETRY TEST ONCE?
4847	023320	041505	042524	020104			
4848	023326	040503	044103	020105			
4849	023334	040520	044522	054524			

4850	023342	042440	051122	051117
4851	023350	052040	051110	020125
4852	023356	040503	044103	042526
4853	023364	026103	053440	046111
4854	023372	020114	042522	051124
4855	023400	020131	042524	052123
4856	023406	047440	041516	000105
4857	023414	047125	054105	042520
4858	023422	052103	042105	046440
4859	023430	044501	020116	042515
4860	023436	047515	054522	050040
4861	023444	051101	052111	020131
4862	023452	051105	047522	020122
4863	023460	044124	052522	041440
4864	023466	041501	053110	041505
4865	023474	020054	044527	046114
4866	023502	051040	052105	054522
4867	023510	052040	051505	020124
4868	023516	047117	042503	000
4869	023523	125	042516	050130
4870	023530	041505	042524	020104
4871	023536	042515	047515	054522
4872	023544	046440	047101	043501
4873	023552	046505	047105	020124
4874	023560	051124	050101	020054
4875	023566	042515	047515	054522
4876	023574	046440	047101	043501
4877	023602	046505	047105	020124
4878	023610	052123	052101	051525
4879	023616	051040	043505	051511
4880	023624	042524	051522	000
4881	023631	123	046525	040515
4882	023636	054522	047440	020106
4883	023644	040515	020120	042522
4884	023652	044507	052123	051105
4885	023660	020123	044124	052101
4886	023666	052040	046511	042105
4887	023674	047440	052125	047440
4888	023702	020116	042522	042101
4889	023710	000		
4890	023711	123	046525	040515
4891	023716	054522	047440	020106
4892	023724	040503	044103	020105
4893	023732	042522	044507	052123
4894	023740	051105	020123	044124
4895	023746	052101	052040	046511
4896	023754	042105	047440	052125
4897	023762	047440	020116	042522
4898	023770	042101	000	
4899	023773	123	046525	040515
4900	024000	054522	047440	020106
4901	024006	040515	020120	042522
4902	024014	044507	052123	051105
4903	024022	020123	047516	020124
4904	024030	047510	042114	047111
4905	024036	020107	042532	047522

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?

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

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

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

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

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

5186	027146	020124	042523	052124	
5187	027154	047111	020107	046101	
5188	027152	047514	051527	047440	
5189	027170	046116	035131	000	
5190	027175	115	050101	051040	EM36: .ASCIZ 'MAP REGISTER UNDER TEST DID NOT RESPOND IN DUAL MAPPING TEST'
5191	027202	043505	051511	042524	
5192	027210	020122	047125	042504	
5193	027216	020122	042524	052123	
5194	027224	042040	042111	047040	
5195	027232	052117	051040	051505	
5196	027240	047520	042116	044440	
5197	027246	020116	052504	046101	
5198	027254	046440	050101	044520	
5199	027262	043516	052040	051505	
5200	027270	000124			
5201	027272	052523	046515	051101	EM37: .ASCIZ 'SUMMARY OF UNIBUS ADDRESS ERRORS, WITH THE MAP RELOCATION DISABLED'
5202	027300	020131	043117	052440	
5203	027306	044516	052502	020123	
5204	027314	042101	051104	051505	
5205	027322	020123	051105	047522	
5206	027330	051522	020054	044527	
5207	027336	044124	052040	042510	
5208	027344	046440	050101	051040	
5209	027352	046105	041517	052101	
5210	027360	047511	020116	044504	
5211	027366	040523	046102	042105	
5212	027374	000			
5213	027375	115	044501	020116	EM40: .ASCIZ 'MAIN MEMORY TIMEOUT OVER THE UNIBUS DID NOT OCCUR PROPERLY'
5214	027402	042515	047515	054522	
5215	027410	052040	046511	047505	
5216	027416	052125	047440	042526	
5217	027424	020122	044124	020105	
5218	027432	047125	041111	051525	
5219	027440	042040	042111	047040	
5220	027446	052117	047440	041503	
5221	027454	051125	050040	047522	
5222	027462	042520	046122	000131	
5223	027470	042522	047514	040503	EM41: .ASCIZ 'RELOCATION THRU THE MAP WAS NOT CORRECT, FULL ADD'
5224	027476	044524	047117	052040	
5225	027504	051110	020125	044124	
5226	027512	020105	040515	020120	
5227	027520	040527	020123	047516	
5228	027526	020124	047503	051122	
5229	027534	041505	026124	043040	
5230	027542	046125	020114	042101	
5231	027550	000104			
5232	027552	042522	047514	040503	EM42: .ASCIZ 'RELOCATION THRU THE MAP WAS NOT CORRECT, CARRY PROPAGATION'
5233	027560	044524	047117	052040	
5234	027566	051110	020125	044124	
5235	027574	020105	040515	020120	
5236	027602	040527	020123	047516	
5237	027610	020124	047503	051122	
5238	027616	041505	026124	041440	
5239	027624	051101	054522	050040	
5240	027632	047522	040520	040507	
5241	027640	044524	047117	000	



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

5298	030344	047522	040520	040507	
5299	030352	044524	047117	200	
5300	030357	124	051505	020124	.ASCIZ ?TEST CODE BEING RUN OVER UNIBUS?
5301	030364	047503	042504	041040	
5302	030372	044505	043516	051040	
5303	030400	047125	047440	042526	
5304	030406	020122	047125	041111	
5305	030414	051525	000		
5306	030417	124	042510	052040	EM50: .ASCII ?THE TOP OF MEMORY IS DIFFERENT THAN THE SIZE JUMPERS?<CRLF>
5307	030424	050117	047440	020106	
5308	030432	042515	047515	054522	
5309	030440	044440	020123	044504	
5310	030446	043106	051105	047105	
5311	030454	020124	044124	047101	
5312	030462	052040	042510	051440	
5313	030470	055111	020105	052512	
5314	030476	050115	051105	100123	
5315	030504	042524	052123	041440	.ASCIZ ?TEST CODE BEING RUN OVER UNIBUS?
5316	030512	042117	020105	042502	
5317	030520	047111	020107	052522	
5318	030526	020116	053117	051105	
5319	030534	052440	044516	052502	
5320	030542	000123			
5321	030544	040520	044522	054524	EM51: .ASCII ?PARITY REPORTING THRU THE MAP IS NOT CORRECT?<CRLF>
5322	030552	051040	050105	051117	
5323	030560	044524	043516	052040	
5324	030566	051110	020125	044124	
5325	030574	020105	040515	020120	
5326	030602	051511	047040	052117	
5327	030610	041440	051117	042522	
5328	030616	052103	200		
5329	030621	124	051505	020124	.ASCIZ ?TEST CODE BEING RUN OVER UNIBUS?
5330	030626	047503	042504	041040	
5331	030634	044505	043516	051040	
5332	030642	047125	047440	042526	
5333	030650	020122	047125	041111	
5334	030656	051525	000		
5335	030661	123	046525	040515	EM52: .ASCIZ ?SUMMARY OF DUAL MAPPING ERRORS?
5336	030666	054522	047440	020106	
5337	030674	052504	046101	046440	
5338	030702	050101	044520	043516	
5339	030710	042440	051122	051117	
5340	030716	000123			
5341	030720	044124	020105	047506	EM201: .ASCIZ ?THE FOLLOWING REGISTERS TIMED OUT WHEN READ?
5342	030726	046114	053517	047111	
5343	030734	020107	042522	044507	
5344	030742	052123	051105	020123	
5345	030750	044524	042515	020104	
5346	030756	052517	020124	044127	
5347	030764	047105	051040	040505	
5348	030772	000104			
5349	030774	044124	020105	047506	EM202: .ASCIZ ?THE FOLLOWING MAP REGISTERS WILL NOT CLEAR?
5350	031002	046114	053517	047111	
5351	031010	020107	040515	020120	
5352	031016	042522	044507	052123	
5353	031024	051105	020123	044527	

052117	051101	043304	042514	041740	040974	040208	039442	038676	037910	037144	036378	035612	034846	034080	033314	032548	031782	031016	030250	029484	028718	027952	027186	026420	025654	024888	024122	023356	022590	021824	021058	020292	019526	018760	018000	017234	016468	015702	014936	014170	013404	012638	011872	011106	010340	009574	008808	008042	007276	006510	005744	004978	004212	003446	002680	001914	001148	000382	000000
--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------

- EM203: .ASCII 'THE FOLLOWING ARE DUAL ADDRESSING ERRORS IN THE UNIBUS MAP'
  
- EM204: .ASCII 'THE COUNT PATTERN THRU THE MAP REGISTERS FAILED'
  
- EM205: .ASCII 'UNIBUS DATA PATH COUNT PATTERN FAILURE'
  
- EM206: .ASCII 'UNIBUS ADDRESSING ERRORS, MAP RELOCATION DISABLED'
  
- EM207: .ASCII 'COUNT PATTERN FAILURES IN CACHE REGISTERS'
  
- CM1: .ASCII 'RECEIVED EXPECTED TESTNO PC AT ABORT'
  
- CM2: .ASCII 'RECEIVED TESTNO PC AT ABORT'





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

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

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



5690	034470	001230	001254	001262			
5691	034476	001116	000000				
5692	034502	001160	001154	001262	DT12:	.WORD	\$REG2, \$REG0, TESTNO, \$ERRPC, 0
5693	034510	001116	000000				
5694	034514	001162	001156	001262	DT13:	.WORD	\$REG3, \$REG1, TESTNO, \$ERRPC, 0
5695	034522	001116	000000				
5696	034526	001226	001224	001232	DT14:	.WORD	ADDROR, ADRAND, DATAOR, DATAND, ERRCNT, TESTNO, 0
5697	034534	001230	001254	001262			
5698	034542	000000					
5699	034544	001226	001224	001236	DT15:	.WORD	ADDROR, ADRAND, PATTOR, PATAND, DATAOR, DATAND, ERRCNT, TESTNO, 0
5700	034552	001234	001232	001230			
5701	034560	001254	001262	000000			
5702	034566	001154	001262	001116	DT17:	.WORD	\$REG0, TESTNO, \$ERRPC, 0
5703	034574	000000					
5704	034576	001156	001262	001116	DT20:	.WORD	\$REG1, TESTNO, \$ERRPC, 0
5705	034604	000000					
5706	034606	001236	001234	001232	DT23:	.WORD	PATTOR, PATAND, DATAOR, DATAND, ERRCNT, TESTNO, 0
5707	034614	001230	001254	001262			
5708	034622	000000					
5709	034624	001164	001154	001262	DT25:	.WORD	\$REG4, \$REG0, TESTNO, \$ERRPC, 0
5710	034632	001116	000000				
5711	034636	001156	001262	001116	DT27:	.WORD	\$REG1, TESTNO, \$ERRPC, 0
5712	034644	000000					
5713	034646	001262	001116	000000	DT30:	.WORD	TESTNO, \$ERRPC, 0
5714	034654	001254	001256	001262	DT34:	.WORD	ERRCNT, CNTR, TESTNO, \$ERRPC, 0
5715	034662	001116	000000				
5716	034666	001240	001242	001262	DT35:	.WORD	LOWEST, HIGEST, TESTNO, \$ERRPC, 0
5717	034674	001116	000000				
5718	034700	001262	001116	001154	DT36:	.WORD	TESTNO, \$ERRPC, \$REG0, 0
5719	034706	000000					
5720	034710	001226	001224	001232	DT37:	.WORD	ADDROR, ADRAND, DATAOR, DATAND, ERRCNT, TESTNO, 0
5721	034716	001230	001254	001262			
5722	034724	000000					
5723	034726	001264	001266	001262	DT40:	.WORD	CPUEXP, PCPLER, TESTNO, \$ERRPC, 0
5724	034734	001116	000000				
5725	034740	001160	001156	001262	DT41:	.WORD	\$REG2, \$REG1, TESTNO, \$ERRPC, 0
5726	034746	001116	000000				
5727	034752	001156	001162	001160	DT42:	.WORD	\$REG1, \$REG3, \$REG2, TESTNO, \$ERRPC, 0
5728	034760	001262	001116	000000			
5729	034766	177760	001320	001262	DT43:	.WORD	SIZEL0, RSIZE, TESTNO, \$ERRPC, 0
5730	034774	001116	000000				
5731	035000	001200	001274	001270	DT44:	.WORD	\$TMP4, PPARER, PLOADR, PMAINT, PCONTR, TESTNO, \$ERRPC, 0
5732	035006	001300	001276	001262			
5733	035014	001116	000000				
5734	035020	001154	001262	001116	DT201:	.WORD	\$REG0, TESTNO, \$ERRPC, 0
5735	035026	000000					
5736	035030	001154	001170	001262	DT202:	.WORD	\$REG0, \$TMP0, TESTNO, \$ERRPC, 0
5737	035036	001116	000000				
5738	035042	001154	001156	001262	DT203:	.WORD	\$REG0, \$REG1, TESTNO, \$ERRPC, 0
5739	035050	001116	000000				
5740	035054	001154	001160	001164	DT204:	.WORD	\$REG0, \$REG2, \$REG4, \$REG3, TESTNO, \$ERRPC, 0
5741	035062	001162	001262	001116			
5742	035070	000000					
5743	035072	001156	001154	001160	DT205:	.WORD	\$REG1, \$REG0, \$REG2, TESTNO, \$ERRPC, 0
5744	035100	001262	001116	000000			
5745	035106	001154	001162	001262	DT206:	.WORD	\$REG0, \$REG3, TESTNO, \$ERRPC, 0

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

5802	035330	000	000	003	DF205:	.BYTE	0,0,3,0,0
5803	035333	000	000				
5804	035335	000	000	000	DF206:	.BYTE	0,0,0,0
5805	035340	000					
5806	035341	000	000	000	DF207:	.BYTE	0,0,0,0,0
5807	035344	000	000				
5808						.EVEN	
5809						.END	
58:0	000001						



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

DH5	031655	1033	5428#				
DH6	031761	1040	1047	5441#			
DISPLA=	177570	474#	1424*	1455*	3209*	3417*	4208*
DT1	034360	1006	5676#				
DT10	034462	1056	1063	5689#			
DT12	034502	1070	5692#				
DT13	034514	1077	5694#				
DT14	034526	1084	5696#				
DT15	034544	1091	1098	5699#			
DT17	034566	1105	1119	1126	5702#		
DT2	034372	1012	5678#				
DT20	034576	1112	5704#				
DT201	035020	1309	5734#				
DT202	035030	1315	5736#				
DT203	035042	1322	5738#				
DT204	035054	1328	5740#				
DT205	035072	1334	5743#				
DT206	035106	1341	5745#				
DT207	035120	1348	5747#				
DT23	034606	1133	1140	1175	5706#		
DT25	034624	1147	1154	5709#			
DT27	034636	1161	5711#				
DT3	034402	1020	1028	5680#			
DT30	034646	1168	1181	1189	5713#		
DT34	034654	1196	5714#				
DT35	034666	1205	5716#				
DT36	034700	1211	5718#				
DT37	034710	1218	1299	5720#			
DT4	034420	5683#					
DT40	034726	1225	1260	5723#			
DT41	034740	1232	1268	5725#			
DT42	034752	1239	1276	5727#			
DT43	034766	1245	1283	5729#			
DT44	035000	1252	1292	5731#			
DT5	034432	1035	5685#				
DT6	034446	1042	1049	5687#			
DUALAD	005276	2191#	2840	3525			
EMTVEC=	000030	577#	2445*	2446*			
EM1	023162	1004	4830#				
EM10	023773	1053	4699#				
EM11	024064	1060	4909#				
EM12	024156	1067	4919#				
EM13	024244	1074	4928#				
EM14	024336	1081	4938#				
EM15	024431	1088	4948#				
EM16	024535	1095	4960#				
EM17	024640	1102	4972#				
EM2	023244	1010	4839#				
EM20	024762	1109	4987#				
EM201	030720	1307	5341#				
EM202	030774	1313	5349#				
EM203	031047	1319	5357#				
EM204	031142	1326	5367#				
EM205	031222	1332	5375#				
EM206	031271	1338	5382#				
EM207	031353	1345	5391#				



Symbol	Description	Symbol	Description	Symbol	Description	Symbol	Description	Symbol	Description
0000	...	0001	...	0002	...	0003	...	0004	...
0005	...	0006	...	0007	...	0008	...	0009	...
0010	...	0011	...	0012	...	0013	...	0014	...
0015	...	0016	...	0017	...	0018	...	0019	...
0020	...	0021	...	0022	...	0023	...	0024	...
0025	...	0026	...	0027	...	0028	...	0029	...
0030	...	0031	...	0032	...	0033	...	0034	...
0035	...	0036	...	0037	...	0038	...	0039	...
0040	...	0041	...	0042	...	0043	...	0044	...
0045	...	0046	...	0047	...	0048	...	0049	...
0050	...	0051	...	0052	...	0053	...	0054	...
0055	...	0056	...	0057	...	0058	...	0059	...
0060	...	0061	...	0062	...	0063	...	0064	...
0065	...	0066	...	0067	...	0068	...	0069	...
0070	...	0071	...	0072	...	0073	...	0074	...
0075	...	0076	...	0077	...	0078	...	0079	...
0080	...	0081	...	0082	...	0083	...	0084	...
0085	...	0086	...	0087	...	0088	...	0089	...
0090	...	0091	...	0092	...	0093	...	0094	...
0095	...	0096	...	0097	...	0098	...	0099	...



POP107	170236	81	845						
POP108	170237	82							
POP109	170238	83							
POP110	170239	84							
POP111	170240	85							
POP112	170241	86							
POP113	170242	87							
POP114	170243	88							
POP115	170244	89							
POP116	170245	90							
POP117	170246	91							
POP118	170247	92							
POP119	170248	93							
POP120	170249	94							
POP121	170250	95							
POP122	170251	96							
POP123	170252	97							
POP124	170253	98							
POP125	170254	99							
POP126	170255	00							
POP127	170256	01							
POP128	170257	02							
POP129	170258	03							
POP130	170259	04							
POP131	170260	05							
POP132	170261	06							
POP133	170262	07							
POP134	170263	08							
POP135	170264	09							
POP136	170265	10							
POP137	170266	11							
POP138	170267	12							
POP139	170268	13							
POP140	170269	14							
POP141	170270	15							
POP142	170271	16							
POP143	170272	17							
POP144	170273	18							
POP145	170274	19							
POP146	170275	20							
POP147	170276	21							
POP148	170277	22							
POP149	170278	23							
POP150	170279	24							
POP151	170280	25							
POP152	170281	26							
POP153	170282	27							
POP154	170283	28							
POP155	170284	29							
POP156	170285	30							
POP157	170286	31							
POP158	170287	32							
POP159	170288	33							
POP160	170289	34							
POP161	170290	35							
POP162	170291	36							
POP163	170292	37							
POP164	170293	38							
POP165	170294	39							
POP166	170295	40							
POP167	170296	41							
POP168	170297	42							
POP169	170298	43							
POP170	170299	44							
POP171	170300	45							
POP172	170301	46							
POP173	170302	47							
POP174	170303	48							
POP175	170304	49							
POP176	170305	50							
POP177	170306	51							
POP178	170307	52							
POP179	170308	53							
POP180	170309	54							
POP181	170310	55							
POP182	170311	56							
POP183	170312	57							
POP184	170313	58							
POP185	170314	59							
POP186	170315	60							
POP187	170316	61							
POP188	170317	62							
POP189	170318	63							
POP190	170319	64							
POP191	170320	65							
POP192	170321	66							
POP193	170322	67							
POP194	170323	68							
POP195	170324	69							
POP196	170325	70							
POP197	170326	71							
POP198	170327	72							
POP199	170328	73							
POP200	170329	74							
POP201	170330	75							
POP202	170331	76							
POP203	170332	77							
POP204	170333	78							
POP205	170334	79							
POP206	170335	80							
POP207	170336	81							
POP208	170337	82							
POP209	170338	83							
POP210	170339	84							
POP211	170340	85							
POP212	170341	86							
POP213	170342	87							
POP214	170343	88							
POP215	170344	89							
POP216	170345	90							
POP217	170346	91							
POP218	170347	92							
POP219	170348	93							
POP220	170349	94							
POP221	170350	95							
POP222	170351	96							
POP223	170352	97							
POP224	170353	98							
POP225	170354	99							
POP226	170355	00							
POP227	170356	01							
POP228	170357	02							
POP229	170358	03							
POP230	170359	04							
POP231	170360	05							
POP232	170361	06							
POP233	170362	07							
POP234	170363	08							
POP235	170364	09							
POP236	170365	10							
POP237	170366	11							
POP238	170367	12							
POP239	170368	13							
POP240	170369	14							
POP241	170370	15							
POP242	170371	16							
POP243	170372	17							
POP244	170373	18							
POP245	170374	19							
POP246	170375	20							
POP247	170376	21							
POP248	170377	22							
POP249	170378	23							
POP250	170379	24							
POP251	170380	25							
POP252	170381	26							
POP253	170382	27							
POP254	170383	28							
POP255	170384	29							
POP256	170385	30							
POP257	170386	31							
POP258	170387	32							
POP259	170388	33							
POP260	170389	34							
POP261	170390	35							
POP262	170391	36							
POP263	170392	37							
POP264	170393	38							
POP265	170394	39							
POP266	170395	40							
POP267	170396	41							
POP268	170397	42							
POP269	170398	43							
POP270	170399	44							
POP271	170400	45							
POP272	170401	46							
POP273	170402	47							
POP274	170403	48							
POP275	170404	49							
POP276	170405	50							
POP277	170406	51							
POP278	170407	52							
POP279	170408	53							
POP280	170409	54							
POP281	170410	55							
POP282	170411	56							
POP283	170412	57							
POP284	170413	58							
POP285	170414	59							
POP286	170415	60							
POP287	170416	61							
POP288	170417	62							
POP289	170418	63							
POP290	170419	64							
POP291	170420	65							
POP292	170421	66							
POP293	170422	67							
POP294	170423	68							
POP295	170424	69							
POP296	170425	70							
POP297	170426	71							
POP298	170427	72							
POP299	170428	73							
POP300	170429	74							
POP301	170430	75							
POP302	170431	76							
POP303	170432	77							
POP304	170433	78							
POP305	170434	79							
POP306	170435	80							
POP307	170436	81							
POP308	170437	82							
POP309	170438	83							
POP310	170439	84							

















# K10

PDP-11/70 UNIBUS MAP DIAGNOSTIC MACY11 27(732) 20-SEP-76 13:32 PAGE 128  
 DEKBFB.P11 CROSS REFERENCE TABLE -- USER SYMBOLS

\$PASS	001100	898#	1412	1428	3462	4772*	4773*	4784	4824					
\$PWRAD	004612	2003#												
\$PWRDN	004470	1974#	1998	2449										
\$PWRMG	004606	2001#												
\$PWRUP	004536	1983#	1988#											
\$QUES	001214	938#	1493											
\$RDCHR=	*****	U	1964											
\$RDDEC=	*****	U	1964											
\$RDLIN=	*****	U	1964											
\$RDOCT=	*****	U	1964											
\$REGAD	001152	921#												
\$REGO	001154	923#	1447*	5683	5692	5702	5709	5718	5734	5736	5738	5740	5743	5745
		5747												
\$REG1	001156	924#	1448*	5694	5704	5711	5725	5727	5738	5743				
\$REG2	001160	925#	1449*	5692	5725	5727	5740	5743	5747					
\$REG3	001162	926#	1450*	5694	5727	5740	5745	5747						
\$REG4	001164	927#	1451*	5709	5740									
\$REG5	001166	928#	1452*											
\$RESRE	003506	1704#	1965											
\$RTRN	023146	2455	2457*	2462*	4800	4819#								
\$SAVRE	003450	1688#	1964											
\$SAVR6	004622	1982#	1988	1989*	1990*	2007#								
\$SCOPE	002140	1368#	2443											
\$SETUP=	000037	850#	1470	2443	2445	2447	2449	2451	2452	2453	2455	2467	2471	4770
\$STUP =	177777	850#												
\$SVLAD	002414	1388	1419#											
\$SVPC =	000204	882#	887											
\$SWR ::	177400	435#	445	450	451	452	453	454	455	456	457	935	936	937
		1360	1361	1362	1363	1364	1379	1391	1393	1394	1399	1400	1401	1408
		1409	1410	1421	1424	1427	1435	1436	1437	1438	1439	1440	1456	1459
		1466	1470	1476	1493	2452	2453	2455	2467	2468	2521	2553	2591	2626
		2660	2694	2737	2781	2828	2874	2918	2963	3007	3057	3113	3210	3251
		3313	3374	3412	3508	3560	3630	3667	3849	3923	3979	4037	4108	4209
		4249	4435	4507	4566	4627	4698	4761	4771	4798	4811	4824		
\$SWRMK=	000200	435#	457	458	1364	1365	1395	1396						
\$TBIT	023154	2466#	4815*	4824#										
\$TIMES	001204	935#	1408*	1415	1418*	1427	2452*	2737*	2828*	2874*	2918*	2963*	3007*	3113*
		3313#	3412*	3508*	3849*	4435*	4771*							
\$TK8	001140	914#												
\$TKS	001136	913#	2440											
\$TMP0	001170	929#	2170	2171*	2172	2173*	2596*	2630*	2664*	2698*	3510*	3527*	3531	3534*
		5683	5736											
\$TMP1	001172	930#												
\$TMP2	001174	931#												
\$TMP3	001176	932#												
\$TMP4	001200	933#	3924*	3943	3979*	4000	4041*	4070	4112*	4141	4508*	4527	4566*	4587
		4631*	4660	4704*	4733	5731								
\$TMP5	001202	934#	3115*	3163	3927*	3948	3982*	4005	4040*	4076	4111*	4147	4511*	4532
		4569*	4592	4630*	4666	4703*	4739							
\$TN =	000045	435#	445	2507	2518	2521#	2532	2535	2550	2553#	2563	2575	2588	2591#
		2605	2610	2623	2626#	2639	2644	2657	2660#	2673	2678	2691	2694#	2707
		2720	2734	2737#	2764	2778	2781#	2807	2825	2828#	2850	2858	2871	2874#
		2897	2902	2915	2918#	2941	2947	2960	2963#	2986	2991	3004	3007#	3030
		3035	3054	3057#	3090	3110	3113#	3194	3205	3210#	3236	3248	3251#	3292
		3310	3313#	3356	3371	3374#	3395	3409	3412#	3413	3463	3490	3505	3508#
		3541	3547	3557	3560#	3578	3617	3627	3630#	3652	3664	3667#	3835	3846

		3849#	3905	3920	3923#	3950	3961	3976	3979#	4007	4018	4034	4037#	4039
		4078	4089	4105	4108#	4192	4204	4209#	4231	4246	4249#	4421	4432	4435#
		4489	4504	4507#	4534	4545	4563	4566#	4594	4605	4624	4627#	4629	4668
		4679	4699#											
STPB	001144	916#	1777*	1788										
STPFLG	001151	920#	1738	1788										
STPS	001142	915#	1775	1788										
STRAP	004426	1943#	2447											
STRP =	000022	1950#	1960#	1961#	1962#	1963#	1964#	1965#	1966#	1967#	1968#			
STRPAD	004446	1947	1958#											
STSTNM	001102	899#	1359	1397	1419*	1424	1428	1445	1455	1493	3208*	3209	3416*	3417
		4207*	4208	4770*										
STYPBN=	***** U	1964												
STYPDS	004202	1879#	1963											
STYPE	003544	1738#	1950	1959										
STYPEC	003704	1757	1764	1770	1775#	1776								
STYPEX	003752	1781	1783	1786#										
STYPOC	004000	1819#	1960											
STYPON	004014	1818	1821#	1962										
STYPOS	003754	1814#	1961											
SXTSTR	002212	1382#												
SSGET4=	000001	4798#	4803											
SSTRP =	000002	1949#	1960	1961	1962	1963	1964	1965	1966	1967	1968			
SOFILL	004177	1815*	1819*	1829	1864#									
.	= 035346	853#	857	859#	882	883#	885#	887#	895#	941	1427	1428	1493	1610#
		1788	1933#	1985	2006	2055#	2434#	2441	2467	2468	4783#	4824	4826#	5675#









	4567	4573	4576	4585	4632	4635	4645	4705	4708	4719	4767	4768	4770	4771	4795
CLRB	4567	4573	4576	4585	4632	4635	4645	4705	4708	4719	4767	4768	4770	4771	4795
CMFB	1407	1780	1919	4062	4133	4652	4725								
COM	2138	1415	1473	1911	2110	2136	2327	2440	2471	2527	2558	2600	2634	2668	2702
DMC	2744	2755	2787	2792	2799	2802	2838	2842	2845	2883	2888	2990	2927	2932	2934
EMCB	3272	2977	2979	3016	3021	3023	3076	3084	3124	3128	3154	3221	3231	3261	3271
FMCB	3278	3285	3324	3332	3334	3343	3388	3422	3431	3434	3441	3445	3450	3454	3456
INCB	3464	3466	3521	3523	3529	3538	3569	3576	3586	3592	3645	3679	3694	3709	3724
IOY	3739	3754	3769	3784	3799	3814	3829	3873	3877	3891	3897	3943	4000	4070	4141
JMP	4224	4263	4278	4293	4308	4323	4338	4353	4368	4383	4398	4413	4457	4461	4475
JSR	4481	4527	4597	4660	4733										
MCV	1397	1401	1573	1579	1589	1750	1752	1758	1778	1782					
	2141	2143	2167	2169	2171	2173	2194	2196	2198	2200	2220	2222	2224	2226	2228
	2230	2251	2253	2255	2257	2280	2282	2284	2286	3597	3599	3600	3602	4815	
	1528	2038	4774												
	1762	1839	1850												
	4677														
	857	1458	1472	1475	1740	1994	2005	2127	2314	2356	2411				
	1414	1446	1462	1601	1845	1853	1897	1990	2041	2124	2311	2353	2386	2408	2469
	2791	2901	3127	3346	3448	3452	3882	3890	3893	4067	4138	4466	4474	4477	4657
	4730	4772													
	1419	1453	1784												
	468														
	861	2179	2206	2236	2263	2292	3327	3437	4823						
	1468	1583	1594	1599	1640	1661	1757	1764	1770	2598	2632	2666	2700	2832	2840
	2885	2929	2974	3018	3126	3156	3314	3345	3509	3525	4806				
	1375	1376	1377	1378	1384	1385	1387	1390	1395	1405	1417	1418	1420	1421	1424
	1425	1447	1448	1449	1450	1451	1452	1455	1463	1478	1481	1485	1486	1488	1489
	1490	1491	1519	1523	1547	1554	1563	1564	1566	1570	1576	1582	1585	1593	1598
	1606	1607	1624	1632	1637	1638	1639	1642	1647	1655	1658	1659	1660	1663	1666
	1689	1690	1691	1692	1693	1694	1695	1696	1697	1698	1705	1706	1707	1708	1709
	1710	1711	1712	1713	1714	1742	1743	1747	1760	1814	1822	1823	1824	1830	1837
	1855	1856	1857	1858	1859	1880	1881	1882	1883	1884	1895	1886	1891	1894	1914
	1920	1921	1922	1923	1924	1926	1927	1943	1944	1947	1974	1975	1976	1977	1978
	1979	1980	1981	1982	1983	1988	1992	1993	1994	1995	1996	1997	1998	1999	2002
	2029	2030	2031	2032	2033	2034	2037	2042	2075	2092	2108	2132	2133	2134	2135
	2149	2150	2151	2165	2192	2218	2249	2278	2319	2320	2321	2323	2324	2332	2333
	2334	2335	2361	2363	2364	2365	2366	2367	2368	2369	2370	2373	2376	2377	2381
	2382	2387	2389	2391	2392	2393	2394	2416	2417	2418	2419	2420	2421	2424	2425
	2426	2437	2438	2442	2443	2444	2445	2446	2447	2448	2449	2450	2451	2455	2456
	2457	2458	2460	2462	2465	2467	2468	2477	2479	2480	2482	2483	2484	2485	2492
	2493	2494	2495	2496	2497	2498	2519	2521	2522	2523	2525	2529	2551	2553	2554
	2556	2560	2589	2591	2592	2593	2596	2602	2624	2626	2627	2630	2636	2658	2660
	2661	2664	2670	2692	2694	2695	2698	2704	2735	2737	2740	2742	2743	2752	2754
	2755	2761	2779	2783	2785	2786	2795	2797	2798	2805	2826	2828	2829	2830	2831
	2835	2836	2847	2872	2874	2876	2877	2879	2881	2882	2894	2916	2918	2920	2921
	2923	2925	2926	2938	2961	2963	2965	2966	2968	2970	2971	2983	3005	3007	3009
	3010	3012	3014	3015	3027	3055	3057	3063	3066	3069	3072	3073	3075	3079	3080
	3082	3087	3111	3113	3115	3118	3120	3122	3123	3139	3140	3141	3142	3143	3144
	3145	3147	3148	3150	3151	3162	3163	3173	3175	3176	3177	3178	3179	3180	3181
	3182	3183	3184	3185	3186	3187	3188	3189	3190	3191	3192	3206	3207	3208	3209
	3211	3212	3213	3214	3215	3216	3217	3220	3223	3228	3233	3249	3251	3252	3253
	3254	3255	3256	3259	3267	3268	3270	3274	3275	3277	3281	3282	3284	3288	3311
	3313	3315	3316	3317	3320	3328	3330	3331	3338	3339	3341	3342	3352	3372	3374
	3375	3376	3379	3380	3383	3387	3391	3410	3412	3414	3415	3416	3417	3418	3419
	3420	3425	3426	3427	3428	3430	3438	3439	3444	3449	3473	3474	3481	3483	3484





.ASCIZ	937	940	1609	2008	2476	4783	4790	4830	4839	4846	4857	4869	4881	4890	4899
	4909	4919	4928	4938	4948	4960	4979	4995	5013	5031	5039	5049	5060	5082	5095
	5104	5114	5130	5151	5184	5190	5201	5213	5223	5232	5242	5251	5269	5284	5300
	5315	5329	5335	5341	5349	5357	5367	5375	5382	5391	5400	5407	5418	5433	5444
	5457	5470	5482	5498	5509	5520	5528	5534	5539	5542	5548	5554	5567	5576	5586
	5593	5606	5616	5620	5629	5635	5644	5655	5666						
.BLKB	2055														
.BLKW	1933														
.BYTE	899	900	905	906	917	918	919	920	1861	1862	1863	1864	4825	5751	5753
	5754	5756	5758	5760	5763	5765	5767	5770	5771	5773	5775	5776	5777	5779	5781
	5782	5784	5786	5788	5790	5792	5795	5796	5798	5800	5802	5804	5806		
.DSABL	435														
.ENABL	435														
.END	5810														
.ENDC	440	454	456	457	458	459	848	850	862	863	885	887	889	896	921
	929	935	936	937	938	986	987	1354	1360	1365	1379	1381	1392	1395	1398
	1399	1401	1403	1410	1414	1419	1424	1427	1428	1429	1435	1453	1459	1463	1469
	1470	1476	1483	1493	1671	1717	1789	1867	1935	1944	1947	1949	1959	1960	1961
	1962	1963	1964	1965	1966	1967	1970	1982	1992	2002	2004	2008	2017	2442	2443
	2445	2447	2449	2451	2452	2453	2455	2467	2469	2471	2473	2476	2508	2509	2516
	2517	2521	2533	2536	2537	2548	2549	2553	2564	2576	2577	2586	2587	2591	2606
	2611	2612	2621	2622	2626	2640	2645	2646	2655	2656	2660	2674	2679	2680	2689
	2690	2694	2708	2721	2722	2732	2733	2737	2738	2765	2766	2776	2777	2781	2808
	2809	2823	2824	2828	2829	2851	2859	2860	2869	2870	2874	2875	2898	2903	2904
	2913	2914	2918	2919	2942	2948	2949	2958	2959	2963	2964	2987	2992	2993	3002
	3003	3007	3008	3031	3036	3037	3052	3053	3057	3091	3092	3108	3109	3113	3114
	3195	3196	3204	3205	3210	3237	3238	3246	3247	3251	3293	3294	3308	3309	3313
	3314	3357	3358	3369	3370	3374	3396	3397	3407	3408	3412	3413	3464	3491	3492
	3503	3504	3508	3509	3542	3548	3549	3555	3556	3560	3579	3618	3619	3625	3626
	3630	3653	3654	3662	3663	3667	3836	3837	3844	3845	3849	3850	3906	3907	3918
	3919	3923	3951	3962	3963	3974	3975	3979	4008	4019	4020	4032	4033	4037	4040
	4079	4090	4091	4103	4104	4108	4111	4150	4193	4194	4203	4204	4209	4232	4233
	4244	4245	4249	4422	4423	4430	4431	4435	4436	4490	4491	4502	4503	4507	4535
	4546	4547	4561	4562	4566	4595	4606	4607	4622	4623	4627	4630	4669	4680	4681
	4696	4697	4698	4703	4742	4754	4757	4759	4761	4762	4764	4770	4776	4779	4780
	4783	4790	4796	4803	4805	4823	4824	4825	4826						
.EQUIV	467	468	470	491	492	493	494	495	496	497	498	499	500	501	530
	531	532	533	534	535	536	537	538	539	558	559	560	561	562	563
	564	565	566	567	619	620	621	622	630	631	632	633	634	635	636
	837	838	839	840	841	842	843	844	845	2071	3210	3850			
.EVEN	1610	2014	2476	4783	4790	4826	5675	5809							
.IF	436	453	454	455	456	457	458	459	848	850	957	862	884	886	889
	895	921	929	935	936	937	941	986	1353	1359	1364	1369	1379	1391	1393
	1394	1395	1399	1400	1401	1410	1412	1421	1426	1427	1428	1434	1445	1456	1459
	1466	1468	1469	1470	1476	1483	1493	1670	1716	1788	1866	1934	1943	1947	1949
	1950	1960	1961	1962	1963	1964	1965	1966	1967	1969	1982	1992	2000	2002	2009
	2016	2437	2442	2443	2445	2447	2449	2451	2452	2453	2455	2467	2470	2471	2472
	2475	2507	2509	2516	2521	2532	2535	2537	2548	2553	2563	2575	2577	2586	2591
	2605	2610	2612	2621	2626	2639	2644	2646	2655	2660	2673	2678	2680	2689	2694
	2707	2720	2722	2732	2737	2738	2764	2766	2776	2781	2807	2809	2823	2829	2829
	2850	2858	2860	2869	2874	2875	2897	2902	2904	2913	2918	2919	2941	2944	2944
	2958	2963	2964	2986	2991	2993	3002	3007	3008	3030	3035	3037	3052	3057	3090
	3092	3108	3113	3114	3194	3196	3204	3210	3236	3238	3246	3251	3292	3294	3309
	3313	3314	3356	3358	3369	3374	3395	3397	3407	3412	3413	3463	3490	3492	3503
	3508	3509	3541	3547	3549	3555	3560	3578	3617	3619	3625	3630	3652	3654	3662
	3667	3835	3837	3844	3849	3850	3905	3907	3918	3923	3950	3961	3963	3974	3979

	4007	4018	4020	4032	4037	4039	4078	4089	4091	4103	4108	4110	4149	4192	4194
	4203	4209	4231	4233	4244	4249	4421	4423	4430	4435	4436	4489	4491	4502	4507
	4534	4545	4547	4561	4566	4594	4605	4607	4622	4627	4629	4668	4679	4681	4696
	4698	4702	4741	4753	4757	4758	4759	4761	4763	4764	4766	4775	4778	4780	4782
	4789	4798	4803	4811	4823	4824	4825								
.IFF	454	456	457	458	863	885	887	889	895	921	987	1354	1392	1395	1401
	1427	1429	1434	1459	1470	1483	1671	1717	1789	1867	1935	1944	1970	2000	2017
	2442	2470	2472	2507	2508	2509	2517	2521	2533	2535	2536	2537	2549	2553	2564
	2575	2576	2577	2587	2591	2606	2610	2611	2612	2622	2626	2640	2644	2645	2646
	2656	2660	2674	2678	2679	2680	2690	2694	2708	2720	2721	2722	2733	2737	2764
	2765	2766	2777	2781	2807	2808	2809	2824	2828	2851	2858	2859	2860	2870	2874
	2899	2902	2903	2904	2914	2918	2942	2947	2948	2949	2959	2963	2987	2991	2992
	2993	3003	3007	3031	3035	3036	3037	3053	3057	3090	3091	3092	3109	3113	3194
	3195	3196	3205	3210	3236	3237	3238	3247	3251	3292	3293	3294	3309	3313	3356
	3357	3358	3370	3374	3395	3396	3397	3408	3413	3464	3490	3491	3492	3504	3508
	3542	3547	3548	3549	3556	3560	3579	3617	3618	3619	3626	3630	3652	3653	3654
	3663	3667	3835	3836	3837	3845	3849	3905	3906	3907	3919	3923	3951	3961	3962
	3963	3975	3979	4008	4018	4019	4020	4033	4037	4040	4079	4089	4090	4091	4104
	4108	4110	4149	4192	4193	4194	4204	4209	4231	4232	4233	4245	4249	4421	4422
	4423	4431	4435	4489	4490	4491	4503	4507	4535	4545	4546	4547	4562	4556	4595
	4605	4606	4607	4623	4627	4630	4669	4680	4681	4697	4698	4702	4741	4754	4763
	4766	4775	4778	4824											
.IFT	1409	1469	2476	4783	4790										
.IFTF	1407	1468	2476	4783	4790										
.IF	435	440	445	450	451	452	454	457	458	459	857	941	1360	1361	1362
	1363	1364	1365	1408	1409	1424	1427	1428	1435	1436	1437	1438	1439	1440	1470
	1493	1788	1959	1960	1961	1962	1963	1964	1965	1966	1967	2443	2445	2451	2452
	2453	2467	2468	2471	4759	4770	4771	4785	4792	4824	4826				
.IRP	850	941	1369	1445	1483	1689	1709	1880	1920	1976	1992	2507	2535	2575	2610
	2644	2678	2720	2764	2807	2858	2902	2947	2991	3035	3090	3194	3236	3292	3356
	3395	3490	3547	3617	3652	3835	3905	3961	4018	4089	4192	4231	4421	4489	4545
	4605	4679	4766												
.LIST	435	457	849	850	857	921	923	924	925	926	927	928	929	930	931
	932	933	934	935	1364	1470	1949	1950	1959	1960	1961	1962	1963	1964	1965
	1966	1967	1968	2471	2476	2507	2521	2535	2553	2575	2591	2610	2626	2644	2660
	2678	2694	2720	2737	2764	2781	2807	2828	2858	2874	2902	2918	2947	2963	2991
	3007	3035	3057	3090	3113	3194	3210	3236	3251	3292	3313	3356	3374	3395	3412
	3490	3508	3547	3560	3617	3630	3652	3667	3835	3849	3905	3923	3961	3979	4018
	4037	4089	4108	4192	4209	4231	4249	4421	4435	4489	4507	4545	4566	4605	4627
	4679	4698	4770	4783	4790	4798									
.MACRO	435	458	888	1950	2455	2507	2535	2575	2610	2644	2678	2720	2764	2807	2858
	2902	2947	2991	3035	3090	3194	3236	3292	3356	3395	3490	3547	3617	3652	3835
	3905	3961	4018	4089	4192	4231	4420	4489	4545	4605	4679				
.MCALL	435	849													
.MLIST	435	457	849	850	857	921	923	924	925	926	927	928	929	930	931
	932	933	934	935	1364	1470	1949	1950	1959	1960	1961	1962	1963	1964	1965
	1966	1967	1968	2471	2476	2507	2521	2535	2553	2575	2591	2610	2626	2644	2660
	2678	2694	2720	2737	2764	2781	2807	2828	2858	2874	2902	2918	2947	2963	2991
	3007	3035	3057	3090	3113	3194	3210	3236	3251	3292	3313	3356	3374	3395	3412
	3490	3508	3547	3560	3617	3630	3652	3667	3835	3849	3905	3923	3961	3979	4018
	4037	4089	4108	4192	4209	4231	4249	4421	4435	4489	4507	4545	4566	4605	4627
	4679	4698	4770	4783	4790	4798									
.PAGE	888	986													
.REM	1														
.REFT	857	923	929												
.SETTL	446	460	585	596	610	759	851	858	864	890	988	1355	1430	1495	1613

	1672	1718	1790	1868	1936	1951	1971	2018	2058	2062	2081	2100	2114	2154	2181
	2208	2239	2266	2295	2297	2340	2399	2432	2433	2507	2535	2575	2610	2644	2678
	2720	2764	2807	2858	2902	2947	2991	3035	3090	3194	3236	3292	3356	3391	3490
	3547	3617	3652	3835	3905	3961	4018	4089	4161	4162	4163	4192	4231	4271	4489
	4545	4605	4679	4755	4829										
.TITLE	435														
.WORD	857	884	886	898	901	902	903	904	907	908	909	910	911	912	921
	923	924	925	926	927	928	929	930	931	932	933	934	941	943	945
	946	947	948	949	950	951	953	955	957	959	961	963	964	965	966
	967	968	969	970	971	972	973	974	975	976	977	978	979	980	981
	982	983	985	1550	1557	1587	1644	1665	1785	1865	2001	2003	2125	2312	2354
	2409	3883	4068	4139	4467	4658	4731	4775	4778	4824	5676	5578	5680	5683	5685
	5697	5689	5692	5694	5696	5699	5702	5704	5706	5709	5711	5713	5714	5716	5718
	5720	5723	5725	5727	5729	5731	5734	5736	5738	5740	5743	5745	5747		

ERRORS DETECTED: 0  
DEFAULT GLOBALS GENERATED: 0

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

