

digital
MADE IN USA

.REM @

IDENTIFICATION

PRODUCT CODE: AC-7979C-MC
PRODUCT NAME: CEKBFC0 PDP-11/70-74MP UNIBUS MAP DIAGNOSTIC
DATE CREATED: MAY, 1979
MAINTAINER: DIAGNOSTIC ENGINEERING
AUTHORS: DALE A. ROEDGER
MODIFIED BY: GEOFFREY A. WHITE
ERNEST M. PREISIG

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

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

COPYRIGHT (C) 1975,1979 BY DIGITAL EQUIPMENT CORPORATION

THE FOLLOWING ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION:

DIGITAL	PDP	UNIBUS	MASSBUS
DEC	DECUS	DECTAPE	DECX/11

TABLE OF CONTENTS

- 1) ABSTRACT
- 2) REQUIREMENTS
 - 2.1 EQUIPMENT
 - 2.2 STORAGE
 - 2.3 PRELIMINARY PROGRAMS
- 3) LOADING PROCEDURE
 - 3.1 METHOD
- 4) STARTING PROCEDURE
 - 4.1 STARTING ADDRESS
 - 4.2 PROGRAM AND OPERATOR ACTION
 - 4.3 SPECIAL STARTING PROCEDURE
- 5) OPERATING PROCEDURE
 - 5.1 OPERATIONAL SWITCH SETTINGS
 - 5.2 SUB-ROUTINE ABSTRACTS
 - 5.3 PROGRAM AND OPERATOR ACTION
- 6) ERRORS
 - 6.1 ERROR HALTS AND DESCRIPTION
 - 6.2 ERROR RECOVERY
 - 6.3 SAMPLE ERROR MESSAGES
- 7) RESTRICTIONS
 - 7.1 STARTING RESTRICTIONS
 - 7.2 OPERATING RESTRICTIONS
- 8) MISCELLANEOUS
 - 8.1 EXECUTION TIME
 - 8.2 ADDRESS GENERATION IN THE PDP-11/70
- 9) PROGRAM DESCRIPTION

1. ABSTRACT

THIS PROGRAM IS DESIGNED TO BE RUN ON A PDP11/70-74MP ON WHICH THE CPU, CACHE, AND MEMORY MANAGEMENT DIAGNOSTIC PROGRAMS HAVE BEEN RUN. THE PROGRAM WILL DETECT ALL ERRORS THAT ORIGINATE WITH THE MAP BOX AND PROVIDE LOOPING CAPABILITIES SO THAT THE FIELD SERVICE ENGINEER CAN VERIFY THE FAILURES. THERE MAY BE SOME CASES, SUCH AS THE CACHE REGISTER DATA PATH, AND CACHE MEMORY DATA PATH, WHERE INTERACTION BETWEEN MODULES PROHIBITS CLOSE ISOLATION, BUT THE FAILING FUNCTION WILL BE CALLED OUT SO THE FIELD SERVICE ENGINEER CAN COMPLETE THE ISOLATION PROCESS. THIS PROGRAM ALSO PROVIDES DIAGNOSTIC COVERAGE FOR THE 'CACHE BYPASS' FUNCTION ON MAP PAGE FOR KB11-CM AND SUBSEQUENT PROCESSORS.

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

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

THE PROGRAM LOADS '044' INTO THE MICRO BREAK REGISTER (17777770) SO THAT A SYNC PULSE IS GENERATED ON PIN # AE1 SLOT10 EVERY TIME A 'NOP' IS EXECUTED. THIS SHOULD HELP IN ISOLATING THE EXACT TIMING OF BAD OR MISSING SIGNALS.

2. REQUIREMENTS

2.1 EQUIPMENT

THE BASIC PDP-11/70-74MP 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 FROM THE UNIBUS.

2.3 PRELIMINARY PROGRAMS

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

3. LOADING PROCEDURE

3.1 METHOD.

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

4. STARTING PROCEDURE

4.1 STARTING ADDRESS

PROGRAM STARTS AT ADDRESS 200

4.2 PROGRAM AND/OR OPERATOR ACTION

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

4.3 SPECIAL STARTING PROCEDURE

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

DEFINITION OF THE BITS IN THE CACHE CONTROL REGISTER:

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

5. OPERATING PROCEDURE

5.1 OPERATIONAL SWITCH SETTINGS

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

5.2 SUB-ROUTINE ABSTRACTS

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

5.2.1 MACRO LIBRARY SUBROUTINES (FOUND IN MOST PROGRAMS)

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

5.2.2 SUBROUTINES UNIQUE TO THIS PROGRAM

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

5.2.3 TRAP AND ABORT HANDLER ROUTINES

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

6. ERRORS

6.1 ERROR HALTS AND DESCRIPTION

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

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

6.2 ERROR RECOVERY

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

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

6.3.1 MULTIPLE TYPE ERRORS

THE FOLLOWING REGISTERS TIMED OUT WHEN REFERENCED

REG.ADR	TESTNO	ERRORPC
170210	000001	015226
170212	000001	015232
170214	000001	015232
170216	000001	015232
170230	000001	015232
170232	000001	015232
170234	000001	015232
170236	000001	015232
170250	000001	015232
170252	000001	015232
170254	000001	015232
170256	000001	015232
170270	000001	015232
170272	000001	015232
170274	000001	015232
170276	000001	015232
170310	000001	015232
170312	000001	015232

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

SUMMARY OF MAP REGISTERS THAT TIMED OUT ON READ

REGADRS REGADRS

OR	AND	#ERRORS	TESTNO	ERRORPC
170376	170210	32	000001	010530

7. RESTRICTIONS

7.1 STARTING RESTRICTIONS

NONE

7.2 OPERATING RESTRICTIONS

NONE

8. MISCELLANEOUS

8.1 EXECUTION TIME

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

8.2 ADDRESS GENERATION IN THE PDP-11/70

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

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

DESCRIPTION OF LINES:

A: VIRTUAL ADDRESS (16 BITS)

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

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

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

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

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

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

9. 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.

@
.END

13	OPERATIONAL SWITCH SETTINGS
27	BASIC DEFINITIONS
152	CACHE REGISTER DEFINITIONS
163	CPU REGISTER DEFINITIONS
177	MEMORY MANAGEMENT DEFINITIONS
326	UNIBUS MAP REGISTER DEFINITIONS
425	TRAP CATCHER
432	STARTING ADDRESS(ES)
438	ACT11 HOOKS
464	COMMON TAGS
569	ERROR POINTER TABLE
970	SCOPE HANDLER ROUTINE
1046	ERROR HANDLER ROUTINE
1111	ERROR MESSAGE TYPE OUT ROUTINE
1229	CONVERT 16-BIT VIRTUAL ADDRESS TO 22-BIT PHYSICAL ADDRESS
1288	SAVE AND RESTORE R0-R5 ROUTINES
1334	TYPE ROUTINE
1407	BINARY TO OCTAL (ASCII) AND TYPE
1485	CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
1553	TRAP DECODER
1568	TRAP TABLE
1588	POWER DOWN AND UP ROUTINES
1635	DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE
1675	***** SUBROUTINES UNIQUE TO THIS PROGRAM *****
1679	TURN OFF AND SAVE T-BIT
1698	RESTORE T-BIT TO ITS PREVIOUS CONDITION
1717	SUBROUTINE TO CLEAR ALL OF THE MAP REGISTERS
1731	SUBROUTINE TO LOG AND REPORT TIMEOUTS OF MAP REGISTERS
1771	SUBROUTINE TO REPORT MAP REGISTERS THAT WILL NOT HOLD ZERO
1798	SUBROUTINE TO REPORT DUAL ADDRESSING WHEN LOADING A MAP REGISTER
1825	SUBROUTINE TO REPORT COUNT PATTERN ERRORS IN MAP REGISTERS
1856	SUBROUTINE TO REPORT COUNT PATTERN ERRORS ON UNIBUS DATA PATH
1883	SUBROUTINE TO REPORT COUNT PATTERN ERRORS IN CACHE REGISTERS
1912	***** TRAP HANDLING ROUTINES *****
1914	CPU TRAP HANDLER ROUTINE
1957	CACHE TRAPS AND ABORTS HANDLER ROUTINE
2016	MEMORY MANAGEMENT TRAPS AND ABORTS HANDLER ROUTINE
2049	*****
2050	***** START OF TEST CODE *****
2209	T1 MAP REGISTER RESPONSE TEST
2238	T2 CACHE REGISTER RESPONSE TEST
2279	T3 CLEAR AND READ LOW 20 REGISTERS LOWER 16 BITS
2315	T4 CLEAR AND READ LOW 20 REGISTERS UPPER 6 BITS
2350	T5 CLEAR AND READ HIGH 20 REGISTERS LOWER 16 BITS
2385	T6 CLEAR AND READ HIGH 20 REGISTERS UPPER 6 BITS
2428	T7 DATA PATH TEST TO MAP REGISTER LOWER 16 BITS
2473	T10 DATA PATH TEST TO MAP REGISTER UPPER 6 BITS
2532	T11 DUAL ADDRESSING ON LOADING AND READING MAP REGISTERS
2584	T12 COUNT PATTERN LOW 20 MAP REGISTERS LOWER 16 BITS
2629	T13 COUNT PATTERN LOW 20 MAP REGISTERS UPPER 6 BITS
2675	T14 COUNT PATTERN HIGH 20 MAP REGISTERS LOWER 16 BITS
2720	T15 COUNT PATTERN HIGH 20 MAP REGISTERS UPPER 6 BITS
2765	T16 CACHE REGISTER DATA PATHS
2824	T17 CACHE REGISTER DATA PATH COUNT PATTERN
2929	T20 MAP REGISTER ADDRESS DECODE TEST
2973	T21 CACHE REGISTER ADDRESS DECODE TEST

3030	T22	DATA PATH, UNIBUS TO MAIN MEMORY
3095	T23	MAP DOESN'T RELOCATE IF NOT ENABLED
3135	T24	SIZE JUMPER LOCATION
3232	T25	ENSURE THAT THERE IS NO DUAL MAPPING
3291	T26	LOAD LOCATIONS 40000 - 137776 WITH THEIR ADDRESSES
3362	T27	MAIN MEMORY TIMEOUT THROUGH MAP
3398	T30	RELOCATION TEST USING LOWEST USABLE MAPPING REG
3582	T31	TEST CARRY PROPAGATION OF MAP'S RELOCATION ADDER
3657	T32	PARITY REPORTING THRU THE MAP, MAIN MEMORY EVEN WORD
3714	T33	PARITY REPORTING THRU THE MAP, MAIN MEMORY ODD WORD
3772	T34	PARITY REPORTING THRU THE MAP, CACHE DATA GROUP 0
3844	T35	PARITY REPORTING THRU THE MAP, CACHE DATA GROUP 1
3917	*****	
3918	THE FOLLOWING TESTS ARE RUN THROUGH THE UNIBUS MAP	
3919	*****	
3948	T36	MAIN MEMORY TIMEOUT THRU MAP, CODE RUN OVER UNIBUS
3989	T37	RELOCATION TEST USING LOWEST USABLE MAPPING REG
4180	T40	TEST CARRY PROPAGATION OF MAP'S RELOCATION ADDER
4253	T41	PARITY REPORTING THRU THE MAP, MAIN MEMORY EVEN WORD
4310	T42	PARITY REPORTING THRU THE MAP, MAIN MEMORY ODD WORD
4371	T43	PARITY REPORTING THRU THE MAP, CACHE DATA GROUP 0
4446	T44	PARITY REPORTING THRU THE MAP, CACHE DATA GROUP 1
4520	T45	CHECK BYP BITS IN UBM
4560	T46	CHECK FOR PRESENCE OF MKA11 REGS. ON KB11-EM OR KB11CM
4602	T47	CHECK CACHE BYPASS ON UNIBUS MAP PAGE
4733	END OF PASS ROUTINE	
4827	ERROR MESSAGES AND DATA TABLES	

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

```
.TITLE PDP-11/70-74MP UNIBUS MAP DIAGNOSTIC
;*COPYRIGHT (C) 1975,1979
;*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-A5).
;*

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

.SBTTL BASIC DEFINITIONS

;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
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
R7= %7           ;;GENERAL REGISTER
```

001100
001100
000700
000600

177776
177774
177772
177570
177570

000011
000012
000015
000200

000000
000001
000002
000003
000004
000005
000006
000007

57		.EQUIV R0,R10	::GENERAL REGISTER
58		.EQUIV R1,R11	::GENERAL REGISTER
59		.EQUIV R2,R12	::GENERAL REGISTER
60		.EQUIV R3,R13	::GENERAL REGISTER
61		.EQUIV R4,R14	::GENERAL REGISTER
62		.EQUIV R5,R15	::GENERAL REGISTER
63	000006	SP=%6	::STACK POINTER
64		.EQUIV SP,KSP	::KERNEL STACK POINTER
65		.EQUIV SP,SSP	::SUPERVISOR STACK POINTER
66		.EQUIV SP,USP	::USER STACK POINTER
67	000007	PC=%7	::PROGRAM COUNTER
68			
69		;*PRIORITY LEVEL DEFINITIONS	
70	000000	PR0= 0	::PRIORITY LEVEL 0
71	000040	PR1= 40	::PRIORITY LEVEL 1
72	000100	PR2= 100	::PRIORITY LEVEL 2
73	000140	PR3= 140	::PRIORITY LEVEL 3
74	000200	PR4= 200	::PRIORITY LEVEL 4
75	000240	PR5= 240	::PRIORITY LEVEL 5
76	000300	PR6= 300	::PRIORITY LEVEL 6
77	000340	PR7= 340	::PRIORITY LEVEL 7
78			
79		;*SWITCH REGISTER SWITCH DEFINITIONS	
80	100000	SW15= 100000	
81	040000	SW14= 40000	
82	020000	SW13= 20000	
83	010000	SW12= 10000	
84	004000	SW11= 4000	
85	002000	SW10= 2000	
86	001000	SW09= 1000	
87	000400	SW08= 400	
88	000200	SW07= 200	
89	000100	SW06= 100	
90	000040	SW05= 40	
91	000020	SW04= 20	
92	000010	SW03= 10	
93	000004	SW02= 4	
94	000002	SW01= 2	
95	000001	SW00= 1	
96		.EQUIV SW09,SW9	
97		.EQUIV SW08,SW8	
98		.EQUIV SW07,SW7	
99		.EQUIV SW06,SW6	
100		.EQUIV SW05,SW5	
101		.EQUIV SW04,SW4	
102		.EQUIV SW03,SW3	
103		.EQUIV SW02,SW2	
104		.EQUIV SW01,SW1	
105		.EQUIV SW00,SW0	
106			
107		;*DATA BIT DEFINITIONS (BIT00 TO BIT15)	
108	100000	BIT15= 100000	
109	040000	BIT14= 40000	
110	020000	BIT13= 20000	
111	010000	BIT12= 10000	
112	004000	BIT11= 4000	

113 002000
114 001000
115 000400
116 000200
117 000100
118 000040
119 000020
120 000010
121 000004
122 000002
123 000001

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

135
136 000004
137 000010
138 000014
139 000014
140 000014
141 000020
142 000024
143 000030
144 000034
145 000060
146 000064
147 000114
148 000240
149 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

151
152
153
154 177740
155 177742
156 177744
157 177746
158 177750
159 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

160
161
162
163
164
165 177760
166
167 177762
168

.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
::CURRENTLY ALL ZERO

169	177764	SYSTID = 177764	::SYSTEM ID REGISTER
170	177766	CPUERR = 177766	::CPU ERROR REGISTER HOLDS CONDITION THAT CAUSED
171			::THE TRAP TO ERRVEC (000004)

.SBTTL MEMORY MANAGEMENT DEFINITIONS

;*MEMORY MANAGEMENT STATUS REGISTER ADDRESSES

181	177572	MMR0=	177572
182	177574	MMR1=	177574
183	177576	MMR2=	177576
184	172516	MMR3=	172516
185		.EQUIV	MMR0,SR0
186		.EQUIV	MMR1,SR1
187		.EQUIV	MMR2,SR2
188		.EQUIV	MMR3,SR3

;*USER 'I' PAGE DESCRIPTOR REGISTERS

192	177600	UIPDR0=	177600
193	177602	UIPDR1=	177602
194	177604	UIPDR2=	177604
195	177606	UIPDR3=	177606
196	177610	UIPDR4=	177610
197	177612	UIPDR5=	177612
198	177614	UIPDR6=	177614
199	177616	UIPDR7=	177616

;*USER 'D' PAGE DESCRIPTOR REGISTORS

203	177620	UDPDR0=	177620
204	177622	UDPDR1=	177622
205	177624	UDPDR2=	177624
206	177626	UDPDR3=	177626
207	177630	UDPDR4=	177630
208	177632	UDPDR5=	177632
209	177634	UDPDR6=	177634
210	177636	UDPDR7=	177636

;*USER 'I' PAGE ADDRESS REGISTERS

214	177640	UIPAR0=	177640
215	177642	UIPAR1=	177642
216	177644	UIPAR2=	177644
217	177646	UIPAR3=	177646
218	177650	UIPAR4=	177650
219	177652	UIPAR5=	177652
220	177654	UIPAR6=	177654
221	177656	UIPAR7=	177656

;*USER 'D' PAGE ADDRESS REGISTERS

222

223

224

225	177660	UDPAR0= 177660
226	177662	UDPAR1= 177662
227	177664	UDPAR2= 177664
228	177666	UDPAR3= 177666
229	177670	UDPAR4= 177670
230	177672	UDPAR5= 177672
231	177674	UDPAR6= 177674
232	177676	UDPAR7= 177676

;*SUPERVISOR 'I' PAGE DESCRIPTOR REGISTERS

233		
234		
235		
236	172200	SIPDR0= 172200
237	172202	SIPDR1= 172202
238	172204	SIPDR2= 172204
239	172206	SIPDR3= 172206
240	172210	SIPDR4= 172210
241	172212	SIPDR5= 172212
242	172214	SIPDR6= 172214
243	172216	SIPDR7= 172216

;*SUPERVISOR 'D' PAGE DESCRIPTOR REGISTERS

244		
245		
246		
247	172220	SDPDR0= 172220
248	172222	SDPDR1= 172222
249	172224	SDPDR2= 172224
250	172226	SDPDR3= 172226
251	172230	SDPDR4= 172230
252	172232	SDPDR5= 172232
253	172234	SDPDR6= 172234
254	172236	SDPDR7= 172236

;*SUPERVISOR 'I' PAGE ADDRESS REGISTERS

255		
256		
257		
258	172240	SIPAR0= 172240
259	172242	SIPAR1= 172242
260	172244	SIPAR2= 172244
261	172246	SIPAR3= 172246
262	172250	SIPAR4= 172250
263	172252	SIPAR5= 172252
264	172254	SIPAR6= 172254
265	172256	SIPAR7= 172256

;*SUPERVISOR 'D' PAGE ADDRESS REGISTERS

266		
267		
268		
269	172260	SDPAR0= 172260
270	172262	SDPAR1= 172262
271	172264	SDPAR2= 172264
272	172266	SDPAR3= 172266
273	172270	SDPAR4= 172270
274	172272	SDPAR5= 172272
275	172274	SDPAR6= 172274
276	172276	SDPAR7= 172276

;*KERNEL 'I' PAGE DESCRIPTOR REGISTERS

277		
278		
279		
280	172300	KIPDR0= 172300

281	172302	KIPDR1= 172302
282	172304	KIPDR2= 172304
283	172306	KIPDR3= 172306
284	172310	KIPDR4= 172310
285	172312	KIPDR5= 172312
286	172314	KIPDR6= 172314
287	172316	KIPDR7= 172316

;*KERNEL 'D' PAGE DESCRIPTOR REGISTERS

290		
291	172320	KDPDR0= 172320
292	172322	KDPDR1= 172322
293	172324	KDPDR2= 172324
294	172326	KDPDR3= 172326
295	172330	KDPDR4= 172330
296	172332	KDPDR5= 172332
297	172334	KDPDR6= 172334
298	172336	KDPDR7= 172336
299		

;*KERNEL 'I' PAGE ADDRESS REGISTERS

300		
301		
302	172340	KIPAR0= 172340
303	172342	KIPAR1= 172342
304	172344	KIPAR2= 172344
305	172346	KIPAR3= 172346
306	172350	KIPAR4= 172350
307	172352	KIPAR5= 172352
308	172354	KIPAR6= 172354
309	172356	KIPAR7= 172356

;*KERNEL 'D' PAGE ADDRESS REGISTERS

310		
311		
312		
313	172360	KDPAR0= 172360
314	172362	KDPAR1= 172362
315	172364	KDPAR2= 172364
316	172366	KDPAR3= 172366
317	172370	KDPAR4= 172370
318	172372	KDPAR5= 172372
319	172374	KDPAR6= 172374
320	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'

321		
322		
323		
324		
325		
326		
327		
328		
329		
330		
331		
332	170200	MAPL00 = 170200
333	170202	MAPH00 = 170202
334	170204	MAPL01 = 170204
335	170206	MAPH01 = 170206
336	170210	MAPL02 = 170210

337	170212	MAPH02 = 170212
338	170214	MAPL03 = 170214
339	170216	MAPH03 = 170216
340	170220	MAPL04 = 170220
341	170222	MAPH04 = 170222
342	170224	MAPL05 = 170224
343	170226	MAPH05 = 170226
344	170230	MAPL06 = 170230
345	170232	MAPH06 = 170232
346	170234	MAPL07 = 170234
347	170236	MAPH07 = 170236
348	170240	MAPL10 = 170240
349	170242	MAPH10 = 170242
350	170244	MAPL11 = 170244
351	170246	MAPH11 = 170246
352	170250	MAPL12 = 170250
353	170252	MAPH12 = 170252
354	170254	MAPL13 = 170254
355	170256	MAPH13 = 170256
356	170260	MAPL14 = 170260
357	170262	MAPH14 = 170262
358	170264	MAPL15 = 170264
359	170266	MAPH15 = 170266
360	170270	MAPL16 = 170270
361	170272	MAPH16 = 170272
362	170274	MAPL17 = 170274
363	170276	MAPH17 = 170276
364	170300	MAPL20 = 170300
365	170302	MAPH20 = 170302
366	170304	MAPL21 = 170304
367	170306	MAPH21 = 170306
368	170310	MAPL22 = 170310
369	170312	MAPH22 = 170312
370	170314	MAPL23 = 170314
371	170316	MAPH23 = 170316
372	170320	MAPL24 = 170320
373	170320	MAPH24 = 170320
374	170324	MAPL25 = 170324
375	170326	MAPH25 = 170326
376	170330	MAPL26 = 170330
377	170332	MAPH26 = 170332
378	170334	MAPL27 = 170334
379	170336	MAPH27 = 170336
380	170340	MAPL30 = 170340
381	170342	MAPH30 = 170342
382	170344	MAPL31 = 170344
383	170346	MAPH31 = 170346
384	170350	MAPL32 = 170350
385	170352	MAPH32 = 170352
386	170354	MAPL33 = 170354
387	170356	MAPH33 = 170356
388	170360	MAPL34 = 170360
389	170362	MAPH34 = 170362
390	170364	MAPL35 = 170364
391	170366	MAPH35 = 170366
392	170370	MAPL36 = 170370

393	170372	MAPH36 = 170372
394	170374	MAPL37 = 170374
395	170376	MAPH37 = 170376
396		.EQUIV MAPL00,MAPL0
397		.EQUIV MAPH00,MAPH0
398		.EQUIV MAPL01,MAPL1
399		.EQUIV MAPH01,MAPH1
400		.EQUIV MAPL02,MAPL2
401		.EQUIV MAPH02,MAPH2
402		.EQUIV MAPL03,MAPL3
403		.EQUIV MAPH03,MAPH3
404		.EQUIV MAPL04,MAPL4
405		.EQUIV MAPH04,MAPH4
406		.EQUIV MAPL05,MAPL5
407		.EQUIV MAPH05,MAPH5
408		.EQUIV MAPL06,MAPL6
409		.EQUIV MAPH06,MAPH6
410		.EQUIV MAPL07,MAPL7
411		.EQUIV MAPH07,MAPH7
412		
413		
414		
415		
416	100000	BYP=BIT15
417	010000	VCIP=BIT12
418	000040	S1=BIT5
419	000020	S0=BIT4
420	000054	S1MOM1=BIT5+BIT3+BIT2
421	000034	S0MOM1=BIT4+BIT3+BIT2
422	177744	MSER=177744
423		
424		.SBTTL TRAP CATCHER
425		.=0
426	000000	
427		;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ``.+2,HALT``
428		;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
429		;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
430		
431		.SBTTL STARTING ADDRESS(ES)
432	000200	.=200
433		
434	000200 000137 010000	JMP @#START ;;JUMP TO STARTING ADDRESS OF PROGRAM
435		;;*****
436		
437		.SBTTL ACT11 HOOKS
438		
439		;*THE FOLLOWING LOCATIONS ARE SETUP TO BE USED WITH ACT11
440		;*
441		;*LOCATION 46 WILL CONTAIN THE ADDRESS OF THE LOGICAL
442		;*END OF THE PROGRAM.
443		;*LOCATION 52 IS USED TO SPECIFY PROGRAM OPERATING REQUIREMENTS
444		;*AND/OR RESTRICTIONS. THIS IS ACCOMPLISHED BY SETTING VARIOUS BITS
445		;*TO A ONE OR A ZERO. THE BITS USED AND THERE MEANING ARE:
446		;*
447		;* BIT 15=1 PROGRAM SHOULD BE POWER FAILED WHILE RUNNING
448		;* =0 NO POWER FAIL DESIRED


```
449
450
451
452
453
454
455      000204
456      000046
457 000046 024566
458      000052
459 000052 000000
460      000204

: *
: *      BIT 14=1 PROGRAM RUN TIME IS MEMORY SIZE DEPENDENT
: *      =0 RUN TIME IS NOT MEMORY SIZE DEPENDENT
: *
: *      BITS 13-0 MUST BE ZERO'S

$SVPC=.      ::SAVE LOCATION COUNTER
.=46         ::SET LOCATION COUNTER
.WORD $ENDAD ::SET LOC.46 TO ADDRESS $ENDAD
.=52         ::SET LOCATION COUNTER
.WORD 0      ::SET LOC.52 TO ZERO
.= $SVPC     ::RESTORE LOCATION COUNTER
```

```
461
462
463
464
465
466
467
468      001100      . =1100
469
470      001100      $CMTAG:      :: START OF COMMON TAGS
471      001100      $PASS: .WORD 0      :: CONTAINS PASS COUNT
472      001102      000      $TSTNM: .BYTE 0      :: CONTAINS THE TEST NUMBER
473      001103      000      $ERFLG: .BYTE 0      :: CONTAINS ERROR FLAG
474      001104      000000      $ICNT: .WORD 0      :: CONTAINS SUBTEST ITERATION COUNT
475      001106      000000      $LPADR: .WORD 0      :: CONTAINS SCOPE LOOP
476      001110      000000      $LPERR: .WORD 0      :: CONTAINS SCOPE RETURN FOR ERRORS
477      001112      000000      $ERTTL: .WORD 0      :: CONTAINS TOTAL ERRORS DETECTED
478      001114      000      $ITEMB: .BYTE 0      :: CONTAINS ITEM CONTROL BYTE
479      001115      001      $ERMAX: .BYTE 1      :: CONTAINS MAX. ERRORS PER TEST
480      001116      000000      $ERRPC: .WORD 0      :: CONTAINS PC OF LAST ERROR INSTRUCTION
481      001120      000000      $GDADR: .WORD 0      :: CONTAINS OF 'GOOD' DATA
482      001122      000000      $BDADR: .WORD 0      :: CONTAINS OF 'BAD' DATA
483      001124      000000      $GDDAT: .WORD 0      :: CONTAINS 'GOOD' DATA
484      001126      000000      $BDDAT: .WORD 0      :: CONTAINS 'BAD' DATA
485      001130      000000 000000 000000      .WORD 0,0,0      :: RESERVED--NOT TO BE USED
486      001136      177560      $TKS: 177560      :: TTY KBD STATUS
487      001140      177562      $TKB: 177562      :: TTY KBD BUFFER
488      001142      177564      $TPS: 177564      :: TTY PRINTER STATUS REG.
489      001144      177566      $TPB: 177566      :: TTY PRINTER BUFFER REG.
490      001146      000      $NULL: .BYTE 0      :: CONTAINS NULL CHARACTER FOR FILLS
491      001147      002      $FILLS: .BYTE 2      :: CONTAINS # OF FILLER CHARACTERS REQUIRED
492      001150      012      $FILLC: .BYTE 12      :: INSERT FILL CHARS. AFTER A 'LINE FEED'
493      001151      000      $TPFLG: .BYTE 0      :: 'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)
494      001152      000000      $REGAD: .WORD 0      :: CONTAINS THE FROM
495      001154      000000      $REG0: .WORD 0      :: WHICH ($REG0) WAS OBTAINED
496      001156      000000      $REG1: .WORD 0      :: CONTAINS (($REGAD)+0)
497      001160      000000      $REG2: .WORD 0      :: CONTAINS (($REGAD)+2)
498      001162      000000      $REG3: .WORD 0      :: CONTAINS (($REGAD)+4)
499      001164      000000      $REG4: .WORD 0      :: CONTAINS (($REGAD)+6)
500      001166      000000      $REG5: .WORD 0      :: CONTAINS (($REGAD)+10)
501      001170      000000      $TMP0: .WORD 0      :: CONTAINS (($REGAD)+12)
502      001172      000000      $TMP1: .WORD 0      :: USER DEFINED
503      001174      000000      $TMP2: .WORD 0      :: USER DEFINED
504      001176      000000      $TMP3: .WORD 0      :: USER DEFINED
505      001200      000000      $TMP4: .WORD 0      :: USER DEFINED
506      001202      000000      $TMP5: .WORD 0      :: USER DEFINED
507      001204      000000      $TIMES: 0      :: MAX. NUMBER OF ITERATIONS
508      001206      000000      $ESCAPE: 0      :: ESCAPE ON ERROR
509      001210      177607 000377      $BELL: .ASCIIZ <207><377><377>      :: CODE FOR BELL
510      001214      077      $QUES: .ASCII /?/      :: QUESTION MARK
511      001215      015      $CRLF: .ASCII <15>      :: CARRIAGE RETURN
512      001216      000012      $LF: .ASCIIZ <12>      :: LINE FEED
513      001220      000000      PADRSL: .WORD 0      :: HOLDS THE LOWER 16 BITS OF A 22 BIT
514      001222      000000      PADRSH: .WORD 0      :: ADDRESS GENERATED FOR TYPE OUT.
515
516
```


Address	Hex	Symbol	Size	Description
517	001224	000000		
518	001224	000000		
519	001226	000000		
520	001230	000000		
521	001232	000000		
522	001234	000000		
523	001236	000000		
524	001240	000000		
525				
526	001242	000000		
527				
528	001244	000000		
529				
530	001246	000000		
531				
532	001250	000000		
533				
534	001252	000000		
535				
536	001254	000000		
537	001256	000000		
538	001260	000000		
539	001262	000000		
540	001264	000000		
541	001266	000000		
542	001270	000000		
543	001272	000000		
544	001274	000000		
545	001276	000000		
546	001300	000000		
547	001302	000000		
548	001304	000000		
549	001306	000000		
550	001310	000000		
551	001312	000000		
552	001314	000000		
553	001316	000000		
554	001320	000000		
555	001322	000000		
556	001324	000000		
557				
558	001326	000200		
559	001330	000		
560	001331	000		
561	001332	000		
562	001333	000		
563				
564				
565				

```

566 ;:*****
567
568 .SBTTL ERROR POINTER TABLE
569
570 ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
571 ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
572 ;*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
573 ;*NOTE1: IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
574 ;*NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
575
576 ;*      EM      ;:POINTS TO THE ERROR MESSAGE
577 ;*      DH      ;:POINTS TO THE DATA HEADER
578 ;*      DT      ;:POINTS TO THE DATA
579 ;*      DF      ;:POINTS TO THE DATA FORMAT
580
581
582 001334 $ERRTB:
583 ;ITEM1
584 001334 024773 EM1 ;NOT THE CORRECT CPU TRAP CONDITION THRU ERRVEC (#004)
585 001336 033615 DH1 ;RECEIVD EXPECTD TESTNO PC AT ABORT
586 001340 036752 DT1 ;PCPUER,CPUEXP,TESTNO,BADPC,0
587 001342 037574 DF1 ; 0, 0, 0, 0
588
589 ;ITEM 2
590 001344 025055 EM2 ;UNEXPECTED CPU TRAP THRU ERRVEC (#004)
591 001346 033661 DH2 ;RECEIVD TESTNO PC AT ABORT
592 001350 036764 DT2 ;PCPUER,TESTNO,BADPC
593 001352 037600 DF2 ; 0, 0, 0
594
595 ;ITEM 3
596 001354 025124 EM3 ;UNEXPECTED CACHE PARITY ERROR THRU CACHVEC (#114)
597 ;WILL RETRY TEST ONCE
598 001356 033715 DH3 ;PARITY ADDRESS MAINTEN CONTROL
599 ;CONDITN REFERENC'D REGISTR REGISTR TESTNO PC AT ABORT
600 001360 036774 DT3 ;PPARER,LOADDR,PMAINT,PCONTR,TESTNO,BADPC,0
601 001362 037603 DF3 ; 0, 2, 0, 0, 0, 0
602
603 ;ITEM 4
604 001364 025225 EM4 ;UNEXPECTED MAIN MEMORY PARITY ERROR THRU CACHVEC (#114)
605 ;WILL RETRY TEST ONCE
606 001366 033715 DH3 ;PARITY ADDRESS MAINTEN CONTROL
607 ;CONDITN REFERENC'D REGISTR REGISTR TESTNO PC AT ABORT
608 001370 036774 DT3 ;PPARER,LOADDR,PMAINT,PCONTR,TESTNO,BADPC,0
609 001372 037603 DF3 ; 0, 2, 0, 0, 0, 0
610
611 ;ITEM 5
612 001374 025334 EM5 ;MEMORY MANAGEMENT TRAP, MEMORY MANAGEMENT STATUS REGISTERS
613 001376 034045 DH5 ;STATUS AUTOI/D VIRTADR
614 ;REGISTR REGISTR REGISTR TESTNO PC AT ABORT
615 001400 037024 DT5 ;PMMR0,PMMR1,PMMR2,TESTNO,BADPC,0
616 001402 037611 DF5 ; 0, 0, 0, 0, 0
617
618 ;ITEM 6
619 001404 025442 EM6 ;SUMMARY OF MAP REGISTERS THAT TIMED OUT ON READ
620 001406 034151 DH6 ;REGADRS REGADRS
621 ; 'OR' 'AND' #ERRORS TESTNO ERRORPC

```


622	001410	037040	DT6	:ADDROR,ADRAND,ERRCNT,TESTNO,\$ERRPC,0
623	001412	037616	DF6	: 0, 0, 1, 0, 0
624				
625			:ITEM 7	
626	001414	025522	EM7	:SUMMARY OF CACHE REGISTERS THAT TIMED OUT ON READ
627	001416	034151	DH6	:REGADRS REGADRS
628				: 'OR' 'AND' #ERRORS TESTNO ERRORPC
629	001420	037040	DT6	:ADDROR,ADRAND,ERRCNT,TESTNO,\$ERRPC,0
630	001422	037616	DF6	: 0, 0, 1, 0, 0
631				
632			:ITEM 10	
633	001424	025604	EM10	:SUMMARY OF MAP REGISTERS NOT HOLDING ZERO IN LOW 16 BITS
634	001426	034241	DH10	:REGADRS REGADRS RECEIVD RECEIVD
635				: 'OR' 'AND' 'OR' 'AND' #ERRORS TESTNO ERRORPC
636	001430	037054	DT10	:ADDROR,ADRAND,DATAOR,DATAND,ERRCNT,TESTNO,\$ERRPC,0
637	001432	037623	DF10	: 0, 0, 0, 0, 1, 0, 0
638				
639			:ITEM 11	
640	001434	025675	EM11	:SUMMARY OF MAP REGISTERS NOT HOLDING ZERO IN UPPER 6 BITS
641	001436	034241	DH10	:REGADRS REGADRS RECEIVD RECEIVD
642				: 'OR' 'AND' 'OR' 'AND' #ERRORS TESTNO ERRORPC
643	001440	037054	DT10	:ADDROR,ADRAND,DATAOR,DATAND,ERRCNT,TESTNO,\$ERRPC,0
644	001442	037623	DF10	: 0, 0, 0, 0, 1, 0, 0
645				
646			:ITEM 12	
647	001444	025767	EM12	:POSSIBLE ERROR IN MAP REGISTER DATA PATH (MAP REG 00)
648	001446	034371	DH12	:COUNT COUNT
649				:EXPECTD RECEIVD TESTNO ERRORPC
650	001450	037074	DT12	:\$REG2,\$REG0,TESTNO,\$ERRPC,0
651	001452	037632	DF12	: 0, 0, 0, 0
652				
653			:ITEM 13	
654	001454	026055	EM13	:NOW PROBABLE ERROR IN MAP REGISTER DATA PATH (MAP REG 20)
655	001456	034371	DH12	:COUNT COUNT
656				:EXPECTD RECEIVD TESTNO ERRORPC
657	001460	037106	DT13	:\$REG3,\$REG1,TESTNO,\$ERRPC,0
658	001462	037632	DF12	: 0, 0, 0, 0
659				
660			:ITEM 14	
661	001464	026147	EM14	:SUMMARY OF DUAL ADDRESSING ERRORS ON LOADING MAP REGISTERS
662	001466	034447	DH14	:REGLOAD REGLOAD REGDUAL REGDUAL
663				: 'OR' 'AND' 'OR' 'AND' #ERRORS TESTNO
664	001470	037120	DT14	:ADDROR,ADRAND,DATAOR,DATAND,ERRCNT,TESTNO,0
665	001472	037636	DF14	: 0, 0, 0, 0, 1, 0
666				
667			:ITEM 15	
668	001474	026242	EM15	:SUMMARY OF COUNT PATTERN FAILURES IN LOWER 16 BITS OF MAP REGIS
669	001476	034566	DH15	:MAPREG MAPREG EXPECTD EXPECTD RECEIVD RECEIVD
670				: 'OR' 'AND' 'OR' 'AND' 'OR' 'AND' #ERRORS TESTNO
671	001500	037136	DT15	:ADDROR,ADRAND,PATTOR,PATAND,DATAOR,DATAND,ERRCNT,TESTNO,0
672	001502	037644	DF15	: 0, 0, 0, 0, 0, 0, 1, 0
673				
674			:ITEM 16	
675	001504	026346	EM16	:SUMMARY OF COUNT PATTERN FAILURES IN UPPER 6 BITS OF MAP REGIST
676	001506	034566	DH15	:MAPREG MAPREG EXPECTD EXPECTD RECEIVD RECEIVD
677				: 'OR' 'AND' 'OR' 'AND' 'OR' 'AND' #ERRORS TESTNO

678	001510	037136	DT15	:ADDROR,ADRAND,PATTOR,PATAND,DATAOR,DATAND,ERRCNT,TESTNO,0
679	001512	037644	DF15	: 0, 0, 0, 0, 0, 1, 0
680				
681			:ITEM 17	
682	001514	026451	EM17	:COULD NOT CLEAR CACHE CONTROL REGISTER
683				:POSSIBLE ERROR IN CACHE REGISTER DATA PATH
684	001516	034745	DH17	:RECEIVD TESTNO ERRORPC
685	001520	037160	DT17	:SREG0,TESTNO,\$ERRPC,0
686	001522	037654	DF17	: 0, 0, 0
687				
688			:ITEM 20	
689	001524	026573	EM20	:COULD NOT CLEAR CACHE MAINTENENCE REGISTER
690				:POSSIBLE ERROR IN CACHE REGISTER DATA PATH
691	001526	034745	DH17	:RECEIVD TESTNO ERRORPC
692	001530	037170	DT20	:SREG1,TESTNO,\$ERRPC,0
693	001532	037654	DF17	: 0, 0, 0
694				
695			:ITEM 21	
696	001534	026721	EM21	:COULD NOT READ 177740 FROM CACHE LO ADDRESS REG (LOADRS)
697				:POSSIBLE ERROR IN CACHE REGISTER DATA PATH
698	001536	034745	DH17	:RECEIVD TESTNO ERRORPC
699	001540	037160	DT17	:SREG0,TESTNO,\$ERRPC,0
700	001542	037654	DF17	: 0, 0, 0
701				
702			:ITEM 22	
703	001544	027065	EM22	:COULD NOT READ 000003 FROM CACHE HI ADDRESS REG (HIADRS)
704				:POSSIBLE ERROR IN CACHE REGISTER DATA PATH
705	001546	034745	DH17	:RECEIVD TESTNO ERRORPC
706	001550	037160	DT17	:SREG0,TESTNO,\$ERRPC,0
707	001552	037654	DF17	: 0, 0, 0
708				
709			:ITEM 23	
710	001554	027231	EM23	:SUMMARY OF COUNT PATTERN FAILURES IN CACHE CONTROL REGISTER
711	001556	034775	DH23	:EXPECTD EXPECTD RECEIVD RECEIVD
712				: 'OR' 'AND' 'OR' 'AND' #ERRORS TESTNO
713	001560	037200	DT23	:PATTOR,PATAND,DATAOR,DATAND,ERRCNT,TESTNO,0
714	001562	037657	DF23	: 0, 0, 0, 0, 1, 0
715				
716			:ITEM 24	
717	001564	027325	EM24	:SUMMARY OF COUNT PATTERN FAILURES IN CACHE MAINTENENCE REGISTER
718	001566	034775	DH23	:EXPECTD EXPECTD RECEIVD RECEIVD
719				: 'OR' 'AND' 'OR' 'AND' #ERRORS TESTNO
720	001570	037200	DT23	:PATTOR,PATAND,DATAOR,DATAND,ERRCNT,TESTNO,0
721	001572	037657	DF23	: 0, 0, 0, 0, 1, 0
722				
723			:ITEM 25	
724	001574	027425	EM25	:REFERENCED MAP REGISTER 0 WITH ADDRESS ONE BIT
725				:DIFFERENT THAN 770200
726	001576	035114	DH25	:ADDRUSED BITDIFF TESTNO ERRORPC
727	001600	037216	DT25	:SREG4,\$REG0,TESTNO,\$ERRPC,0
728	001602	037665	DF25	:3, 4, 0, 0
729				
730			:ITEM 26	
731	001604	027532	EM26	:REFERENCED CACHE LOW ADDRESS REGISTER WITH
732				:ADDRESS ONE BIT DIFFERENT THAN 777740
733	001606	035114	DH25	:ADDRUSED BITDIFF TESTNO ERRORPC

734	001610	037216	DT25	;\$REG4,\$REG0,TESTNO,\$ERRPC,0
735	001612	037665	DF25	:3, 4, 0, 0
736				
737			:ITEM 27	
738	001614	027532	EM26	:REFERENCED CACHE LO ADDRESS REGISTER WITH
739				:ONE BIT DIFFERENT THAN 777740
740	001616	035156	DH27	:ADDRUSED TESTNO ERRORPC
741	001620	037230	DT27	;\$REG1,TESTNO,\$ERRPC
742	001622	037671	DF27	:3, 0, 0
743				
744			:ITEM 30	
745	001624	027653	EM30	:CAN'T GET TO MAIN MEMORY FROM UNIBUS WITH THE MAP OFF
746				:SO I'LL JUMP TO THE SIZE JUMPER TEST FOR VERIFICATION
747	001626	035210	DH30	:TESTNO ERRORPC
748	001630	037240	DT30	:TESTNO,\$ERRPC,0
749	001632	037674	DF30	:0, 0
750				
751			:ITEM 31	
752	001634	030027	EM31	:SUMMARY OF COUNT PATTERN FAILURES ON THE UNIBUS DATA PATH
753	001636	034775	DH23	:EXPECTD EXPECTD RECEIVD RECEIVD
754				: 'OR' 'AND' 'OR' 'AND' #ERRORS TESTNO
755	001640	037200	DT23	:PATTOR,PATAND,DATAOR,DATAND,ERRCTN,TESTNO,0
756	001642	037657	DF23	: 0, 0, 0, 0, 1, 0
757				
758			:ITEM 32	
759	001644	030121	EM32	:UNIBUS MAP IS RELOCATING WHEN NOT ENABLED
760	001646	035210	DH30	:TESTNO ERRORPC
761	001650	037240	DT30	:TESTNO,\$ERRPC,0
762	001652	037674	DF30	: 0, 0
763				
764			:ITEM 33	
765	001654	030173	EM33	:CANNOT USE ANY OF THE MAP REGISTERS OR PHYSICAL
766				:ADDRESS BIT14 IS STUCK LOW, MUST RESTART PROGRAM
767				:IF YOU DON'T LOOP ON THIS PROBLEM.
768	001656	035210	DH30	:TESTNO ERRORPC
769	001660	037240	DT30	:TESTNO,\$ERRPC,0
770	001662	037674	DF30	: 0, 0
771				
772			:ITEM 34	
773	001664	030350	EM34	:THE NUMBER OF MAP REGISTERS REMOVED BY JUMPER SETTING
774				:DOES NOT AGREE WITH THE NUMBER FOUND TO BE MISSING.
775	001666	035230	DH34	:REMOVED MISSING TESTNO ERRORPC
776	001670	037246	DT34	:ERRCNT,CNTR,TESTNO,\$ERRPC,0
777	001672	037676	DF34	: 0, 0, 0, 0
778				
779			:ITEM 35	
780	001674	030521	EM35	:THE SIZE JUMPERS ON THE UNIBUS MAP ARE NOT SET
781				:IN THEIR DEFAULT POSITION. THIS WOULD ALLOW UNIBUS
782				:ADDRESSES 000000 TO 757776 TO REFERNECE MAIN MEMORY
783				:THEIR CURRENT SETTING ALLOWS ONLY:
784	001676	035270	DH35	:LOWEST HIGHEST TESTNO ERRORPC
785	001700	037260	DT35	:LOWEST,HIGEST,TESTNO,\$ERRPC,0
786	001702	037702	DF35	: 4, 4, 0, 0
787				
788			:ITEM 36	
789	001704	031006	EM36	:MAP REGISTER UNDER TEST DID NOT RESPOND IN DUAL MAPPING TEST

790	001706	035330	DH36	:TESTNO ERRORPC UNIBUS ADDRESS OF MAP REGISTER UNDER TEST
791	001710	037272	DT36	:TESTNO,\$ERRPC,\$REG0,0
792	001712	037706	DF36	: 0, 0, 3
793				
794			:ITEM 37	
795	001714	031103	EM37	:SUMMARY OF UNIBUS ADDRESS ERRORS, WITH MAP RELOCATION DISABLED
796	001716	034775	DH23	:EXPECTD EXPECTD RECEIVD RECEIVD
797				: 'OR' 'AND' 'OR' 'AND' #ERRORS TESTNO
798	001720	037302	DT37	:ADDROR,ADRAND,DATAOR,DATAND,ERRCNT,TESTNO
799	001722	037711	DF37	: 0, 0, 0, 0, 1, 0
800				
801			:ITEM 40	
802	001724	031206	EM40	:MAIN MEMORY TIME OUT OVER THE UNIBUS DID NOT OCCUR PROPERLY.
803	001726	035422	DH40	:CONDITN CONDITN
804				:EXPECTD RECEIVD TESTNO ERRORPC
805	001730	037320	DT40	:CPUEXP,PCPUER,TESTNO,\$ERRPC,0
806	001732	037717	DF40	: 0, 0, 0, 0
807				
808			:ITEM 41	
809	001734	031301	EM41	:RELOCATION THROUGH THE MAP WAS NOT CORRECT, FULL ADD.
810	001736	035502	DH41	:CORRECT ADDRESS
811				:ADDRESS FETCHED TESTNO ERRORPC
812	001740	037332	DT41	: \$REG2,\$REG1,TESTNO,\$ERRPC
813	001742	037723	DF41	: 0, 0, 0, 0
814				
815			:ITEM 42	
816	001744	031363	EM42	:RELOCATION THRU THE MAP WAS NOT CORRECT, CARRY PROPAGATION
817	001746	035562	DH42	:CORRECT EXPECTD RECEIVD
818				:ADDRESS DATA FROM UB TESTNO ERRORPC
819	001750	037344	DT42	: \$REG1,\$REG3,\$REG2,TESTNO,\$ERRPC
820	001752	037727	DF42	: 3, 0, 0, 0, 0
821				
822			:ITEM 43	
823	001754	031456	EM43	:THE TOP OF MEMORY IS DIFFERENT THAN THE SIZE JUMPERS
824	001756	035662	DH43	:SIZJUMP TOPFOUND TESTNO ERRORPC
825	001760	037360	DT43	:SIZELO,RSIZE,TESTNO,\$ERRPC,0
826	001762	037734	DF43	: 0, 0, 0, 0
827				
828			:ITEM 44	
829	001764	031543	EM44	:PARITY REPORTING THRU THE MAP IS NOT CORRECT
830	001766	035722	DH44	:CONDITN CONDITN ADDRESS MAINTEN CONTROL
831				:EXPECTD RECEIVD REFERENCED REGISTR REGISTR TESTNO ERRORPC
832	001770	037372	DT44	: \$TMP4,PPARER,PLOADR,PMaint,PContr,TESTNO,\$ERRPC,0
833	001772	037740	DF44	: 0, 0, 2, 0, 0, 0, 0
834				
835			:ITEM 45	
836	001774	031620	EM45	:MAIN MEMORY TIME OUT OVER THE UNIBUS DID NOT OCCUR PROPERLY.
837				:TEST BEING RUN OVER UNIBUS
838	001776	035422	DH40	:CONDITN CONDITN
839				:EXPECTD RECEIVD TESTNO ERRORPC
840	002000	037320	DT40	:CPUEXP,PCPUER,TESTNO,\$ERRPC,0
841	002002	037717	DF40	: 0, 0, 0, 0
842				
843			:ITEM 46	
844	002004	031753	EM46	:RELOCATION THROUGH THE MAP WAS NOT CORRECT, FULL ADD.
845				:TEST BEING RUN OVER UNIBUS

846	002006	035502	DH41	:CORRECT ADDRESS
847				:ADDRESS FETCHED TESTNO ERRORPC
848	002010	037332	DT41	:\$REG2,\$REG1,TESTNO,\$ERRPC
849	002012	037723	DF41	: 0, 0, 0, 0
850				
851			:ITEM 47	
852	002014	032075	EM47	:RELOCATION THRU THE MAP WAS NOT CORRECT, CARRY PROPAGATION
853				:TEST BEING RUN OVER UNIBUS
854	002016	035562	DH42	:CORRECT EXPECTD RECEIVD
855				:ADDRESS DATA FROM UB TESTNO ERRORPC
856	002020	037344	DT42	:\$REG1,\$REG3,\$REG2,TESTNO,\$ERRPC
857	002022	037727	DF42	: 3, 0, 0, 0, 0
858				
859			:ITEM 50	
860	002024	032230	EM50	:THE TOP OF MEMORY IS DIFFERENT THAN THE SIZE JUMPERS
861				:TEST BEING RUN OVER UNIBUS
862	002026	035662	DH43	:SIZJUMP TOPFOUND TESTNO
863	002030	037360	DT43	:SIZELO,RSIZE,TESTNO,0
864	002032	037734	DF43	: 0, 0, 0
865				
866				
867			:ITEM 51	
868	002034	032355	EM51	:PARITY REPORTING THRU THE MAP IS NOT CORRECT
869				:TEST BEING RUN OVER UNIBUS
870	002036	035722	DH44	:CONDITN CONDITN ADDRESS MAINTEN CONTROL
871				:EXPECTD RECEIVD REFERENC'D REGISTR REGISTR TESTNO ERRORPC
872	002040	037372	DT44	:\$TMP4,\$PPARER,\$PLOADR,\$PMAINT,\$PCONTR,TESTNO,\$ERRPC,0
873	002042	037740	DF44	: 0, 0, 2, 0, 0, 0, 0
874				
875			:ITEM 52	
876	002044	032472	EM52	:SUMMARY OF DUAL MAPPING ERRORS
877	002046	034775	DH23	:EXPECTD EXPECTD RECEIVD RECEIVD
878				: 'OR' 'AND' 'OR' 'AND' #ERRORS TESTNO
879	002050	037302	DT37	:ADDROR,ADRAND,DATAOR,DATAND,\$ERRPC,TESTNO,0
880	002052	037711	DF37	: 0, 0, 0, 0, 1, 0
881				
882			:ITEM 53	
883				
884	002054	032531	EM53	:BYP BIT IN UBMR COULD NOT BE CLEARED
885	002056	036066	DH53	
886	002060	037412	DT53	
887	002062	037674	DF30	
888				
889			:ITEM 54	
890				
891	002064	032576	EM54	:BYP BIT IN UBMR COULD NOT BE SET
892	002066	036066	DH53	
893	002070	037412	DT53	
894	002072	037674	DF30	
895				
896			:ITEM 55	
897	002074	032637	EM55	:MK11 CSR COULD NOT BE ACCESSED
898	002076	036115	DH55	
899	002100	037422	DT55	
900	002102	037574	DF1	
901				

```

902      :ITEM 56
903
904      002104 032713      EM56      ;TEST DATA REF NOT A HIT
905      002106 036160      DH56
906      002110 037434      DT56
907      002112 037632      DF12
908
909      :ITEM 57
910
911      002114 032751      EM57      ;TEST DATA REF NOT A MISS
912      002116 036225      DH57      ;CACHE BYPASS ON UNIBUS MAP DID NOT INVALIDATE CACHED DATA
913      002120 037446      DT57
914      002122 037574      DF1
915
916      002124      ER200:      ;THIS IS THE STARTING POINT FOR ERROR MESSAGES
917                                ;201 THRU 377. THEY ARE USED FOR MULTIPLE
918                                ;ERROR MESSAGES.
919
920      :ITEM 201
921      002124 033110      EM201      ;THE FOLLOWING REGISTERS TIMED OUT WHEN READ
922      002126 036271      DH201      ;REGADRS TESTNO ERRORPC
923      002130 037460      DT201      ;$REG0,TESTNO,$ERRPC,0
924      002132 037747      DF201      ; 0, 0, 0
925
926      :ITEM 202
927      002134 033164      EM202      ;THE FOLLOWING MAP REGISTERS WILL NOT CLEAR
928      002136 036321      DH202      ;REGADRS DATAREC TESTNO ERRORPC.
929      002140 037470      DT202      ;$REG0,$TMP0,TESTNO,$ERRPC,0
930      002142 037752      DF202      ; 0, 0, 0, 0
931
932      :ITEM 203
933      002144 033237      EM203      ;THE FOLLOWING ARE DUAL ADDRESSING ERRORS IN THE UNIBUS MAP
934      002146 036361      DH203      ;MAPREG MAPREG
935                                ;TESTING DUALED TESTNO ERRORPC
936      002150 037502      DT203      ;$REG0,$REG1,$TESTNO,$ERRPC,0
937      002152 037756      DF203      ; 0, 0, 0, 0
938
939      :ITEM 204
940      002154 033332      EM204      ;THE COUNT PATTERN THRU THE MAP REGISTERS FAILED
941      002156 036440      DH204      ;REGADRS PATTERN EXPECTD RECEIVD TESTNO ERRORPC
942      002160 037514      DT204      ;$REG0,$REG2,$REG4,$REG3,TESTNO,$ERRPC,0
943      002162 037762      DF204      ; 0, 0, 0, 0, 0, 0
944
945      :ITEM 205
946      002164 033412      EM205      ;UNIBUS DATA PATH COUNT PATTERN FAILURE
947      002166 036520      DH205      ;EXPECTD RECEIVD ADDRSLD TESTNO ERRORPC
948      002170 037532      DT205      ;$REG1,$REG0,$REG2,TESTNO,$ERRPC,0
949      002172 037770      DF205      ;0, 0, 3, 0, 0
950
951      :ITEM 206
952      002174 033461      EM206      ;UNIBUS ADDRESSING ERRORS, MAP RELOCATION DISABLED
953      002176 036572      DH206      ;ADDRESS ADDRESS
954                                ;EXPECTD RECEIVD TESTNO ERRORPC
955      002200 037546      DT206      ;$REG0,$REG3,TESTNO,$ERRPC,0
956      002202 037775      DF206      ; 0, 0, 0, 0
957

```



```

958      :ITEM 207
959      EM207      :COUNT PATTERN FAILURES IN CACHE REGISTERS
960      DH207      :REGISTR EXPECTD RECEIVD
961      :ADDRESS DATA DATA TESTNO ERRORPC
962      DT207      :$REG0,$REG2,$REG3,TESTNO,$ERRPC,0
963      DF207      : 0, 0, 0, 0, 0
964
965
966
967      ::*****
968
969      .SBTTL SCOPE HANDLER ROUTINE
970
971      :*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
972      :*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
973      :*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
974      :*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
975      :*SW14=1      LOOP ON TEST
976      :*SW11=1      INHIBIT ITERATIONS
977      :*SW09=1      LOOP ON ERROR
978      :*SW08=1      LOOP ON TEST IN SWR<6:0>
979      :*CALL
980      :*          SCOPE          ;;SCOPE=IOT
981
982      $SCOPE:
983      002214      005037      C01322      CLR      RETRY      :CLEAR RETRY FLAG AN THE START OF
984      :EACH TEST
985      002220      005037      001254      CLR      ERRCNT      :CLEAR THE MULTIPLE ERROR COUNTER
986      002224      005037      001232      CLR      DATAOR      :LOCATION FOR LOGICAL OR OF BAD DATA
987      002230      005037      001226      CLR      ADDROR      :LOCATION FOR LOGICAL OR OF ADDRESS
988      002234      005037      001236      CLR      PATTOR      :LOCATION FOR LOGICAL OR OF PATTERN LOADED
989      002240      012700      177777      MOV      #-1,R0      :LOAD -1 INTO R0 TO INITIALIZE LOGICAL AND LOCS
990      002244      010037      001230      MOV      R0,DATAND      :LOCATION FOR LOGICAL AND OF BAD DATA
991      002250      010037      001224      MOV      R0,ADRAND      :LOCATION FOR LOGICAL AND OF ADDRESS
992      002254      010037      001234      MOV      R0,PATAND      :LOCATION FOR LOGICAL AND OF PATTERN LOADED
993      002260      006137      177570      ROL      @#SWR      :LOOP ON PRESENT TEST?
994      002264      100514      BMI      $OVER      :YES IF SW14=1
995      :*****START OF CODE FOR THE XOR TESTER*****
996      002266      000416      $XTSTR: BR      6$      :IF RUNNING ON THE 'XOR' TESTER CHANGE
997      :THIS INSTRUCTION TO A 'NOP' (NOP=240)
998      002270      013746      000004      MOV      @#ERRVEC,-(SP) :SAVE THE CONTENTS OF THE ERROR VECTOR
999      002274      012737      002314      000004      MOV      #5$,@#ERRVEC :SET FOR TIMEOUT
1000      002302      005737      177060      TST      @#177060 :TIME OUT ON XOR?
1001      002306      012637      000004      MOV      (SP)+,@#ERRVEC :RESTORE THE ERROR VECTOR
1002      002312      000466      BR      $SVLAD :GO TO THE NEXT TEST
1003      002314      022626      5$: CMP      (SP)+,(SP)+ :CLEAR THE STACK AFTER A TIME OUT
1004      002316      012637      000004      MOV      (SP)+,@#ERRVEC :RESTORE THE ERROR VECTOR
1005      002322      000426      BR      7$ :LOOP ON THE PRESENT TEST
1006      002324      6$:*****END OF CODE FOR THE XOR TESTER*****
1007      002324      032737      000400      177570      BIT      #BIT08,@#SWR :LOOP ON SPEC. TEST?
1008      002332      001407      BEQ      2$ :BR IF NO
1009      002334      013746      177570      MOV      @#SWR,-(SP) :SET DESIRED TEST NUM. FROM SWR
1010      002340      042716      000200      BIC      #5$WRMK,(SP) :STRIP AWAY UNDESIED BITS
1011      002344      122637      001102      CMPB      (SP)+,$TSTNM :ON THE RIGHT TEST?
1012      002350      001462      BEQ      $OVER :BR IF YES
1013      002352      105737      001103      2$: TSTB      $ERFLG :HAS AN ERROR OCCURRED?

```

```
1014 002356 001421 BEQ 3$ ::BR IF NO
1015 002360 123737 001115 001103 CMPB $ERMAX,$ERFLG ::MAX. ERRORS FOR THIS TEST OCCURRED?
1016 002366 101015 BHI 3$ ::BR IF NO
1017 002370 032737 001000 177570 BIT #BIT09,@#SWR ::LOOP ON ERROR?
1018 002376 001404 BEQ 4$ ::BR IF NO
1019 002400 013737 001110 001106 7$: MOV $LPERR,$LPADR ::SET LOOP ADDRESS TO LAST SCOPE
1020 002406 000443 BR $OVER
1021 002410 105037 001103 4$: CLRB $ERFLG ::ZERO THE ERROR FLAG
1022 002414 005037 001204 CLR $TIMES ::CLEAR THE NUMBER OF ITERATIONS TO MAKE
1023 002420 000415 BR 1$ ::ESCAPE TO THE NEXT TEST
1024 002422 032737 004000 177570 3$: BIT #BIT11,@#SWR ::INHIBIT ITERATIONS?
1025 002430 001011 BNE 1$ ::BR IF YES
1026 002432 005737 001100 TST $PASS ::IF FIRST PASS OF PROGRAM
1027 002436 001406 BEQ 1$ :: INHIBIT ITERATIONS
1028 002440 005237 001104 INC $ICNT ::INCREMENT ITERATION COUNT
1029 002444 023737 001204 001104 CMP $TIMES,$ICNT ::CHECK THE NUMBER OF ITERATIONS MADE
1030 002452 002021 BGE $OVER ::BR IF MORE ITERATION REQUIRED
1031 002454 012737 000001 001104 1$: MOV #1,$ICNT ::REINITIALIZE THE ITERATION COUNTER
1032 002462 013737 002532 001204 MOV $MXCNT,$TIMES ::SET NUMBER OF ITERATIONS TO DO
1033 002470 105237 001102 $SVLAD: INCB $TSTNM ::COUNT TEST NUMBERS
1034 002474 011637 001106 MOV (SP),$LPADR ::SAVE SCOPE LOOP ADDRESS
1035 002500 011637 001110 MOV (SP),$LPERR ::SAVE ERROR LOOP ADDRESS
1036 002504 005037 001206 CLR $ESCAPE ::CLEAR THE ESCAPE FROM ERROR ADDRESS
1037 002510 112737 000001 001115 MOVB #1,$ERMAX ::ONLY ALLOW ONE(1) ERROR ON NEXT TEST
1038 002516 013737 001102 177570 $OVER: MOV $TSTNM,@#DISPLAY ::DISPLAY TEST NUMBER
1039 002524 013716 001106 MOV $LPADR,(SP) ::FUDGE RETURN ADDRESS
1040 002530 000002 RTI ::FIXES PS
1041 002532 000144 $MXCNT: 100. ::MAX. NUMBER OF ITERATIONS
1042 ::*****
1043
1044 .SBTTL ERROR HANDLER ROUTINE
1045
1046 :*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT.
1047 :*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
1048 :*AND GO TO ERTYPE ON ERROR
1049 :*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
1050 :*SW15=1 HALT ON ERROR
1051 :* HALT CAN OCCUR BEFORE AND AFTER THE ERROR TYPEOUT
1052 :*SW13=1 INHIBIT ERROR TYPEOUTS
1053 :*SW10=1 BELL ON ERROR
1054 :*SW09=1 LOOP ON ERROR
1055 :*CALL
1056 :* ERROR N ::ERROR=EMT AND N=ERROR ITEM NUMBER
1057
1058 002534 $ERROR:
1059 002534 113737 001102 001262 MOVB $TSTNM,TESTNO ::SAVE TEST NUMBER FOR ERROR TYPE OUT
1060 002542 005237 001254 INC ERRCNT ::COUNT ALL MULTIPLE ERRORS
1061 002546 010037 001154 MOV R0,$REG0 ::SAVE R0 FOR POSSIBLE TYPE OUT
1062 002552 010137 001156 MOV R1,$REG1 ::SAVE R1 FOR POSSIBLE TYPE OUT
1063 002556 010237 001160 MOV R2,$REG2 ::SAVE R2 FOR POSSIBLE TYPE OUT
1064 002562 010337 001162 MOV R3,$REG3 ::SAVE R3 FOR POSSIBLE TYPE OUT
1065 002566 010437 001164 MOV R4,$REG4 ::SAVE R4 FOR POSSIBLE TYPE OUT
1066 002572 010537 001166 MOV R5,$REG5 ::SAVE R5 FOR POSSIBLE TYPE OUT
1067 002576 105237 001103 7$: INCB $ERFLG ::SET THE ERROR FLAG
1068 002602 001775 BEQ 7$ ::DON'T LET THE FLAG GO TO ZERO
1069 002604 013737 001102 177570 MOV $TSTNM,@#DISPLAY ::DISPLAY TEST NUMBER AND ERROR FLAG
```



```
1070 002612 005737 177570      TST      @#SWR      ;;HALT ON ERROR = 1?
1071 002616 100001      BPL      8$      ;;BRANCH IF NO
1072 002620 000000      HALT      ;;YES--HALT
1073 002622 032737 002000 177570 8$:    BIT      #BIT10,@#SWR  ;;BELL ON ERROR?
1074 002630 001402      BEQ      1$      ;;NO - SKIP
1075 002632 104400 001210      TYPE     ,SBELL      ;;RING BELL
1076 002636 005237 001112      INC      $ERTTL      ;;COUNT THE NUMBER OF ERRORS
1077 002642 011637 001116      MOV      (SP), $ERRPC  ;;GET ADDRESS OF ERROR INSTRUCTION
1078 002646 162737 000002 001116      SUB      #2,$ERRPC
1079 002654 117737 176236 001114      MOVB     @ $ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
1080 002662 032737 020000 177570      BIT      #BIT13,@#SWR  ;;SKIP TYPEOUT IF SET
1081 002670 001004      BNE      2$      ;;SKIP TYPEOUTS
1082 002672 004737 003036      JSR      PC,ERTYPE  ;;GO TO USER ERROR ROUTINE
1083 002676 104400 001215      TYPE     , $CRLF
1084 002702 005737 177570      TST      @#SWR      ;;HALT ON ERROR
1085 002706 100001      BPL      9$      ;;SKIP IF CONTINUE
1086 002710 000000      HALT      ;;HALT ON ERROR!
1087 002712 022737 024566 000042 9$:    CMP      # $ENDAD,42  ;;ACT-11?
1088 002720 001001      BNE      3$      ;;BRANCH IF NO
1089 002722 000000      HALT      ;;YES
1090 002724 032737 001000 177570 3$:    BIT      #BIT09,@#SWR  ;;LOOP ON ERROR SWITCH SET?
1091 002732 001402      BEQ      4$      ;;BR IF NO
1092 002734 013716 001110      MOV      $LPERR,(SP)  ;;FUDGE RETURN FOR LOOPING
1093 002740 005737 001206      TST      $ESCAPE  ;;CHECK FOR AN ESCAPE ADDRESS
1094 002744 001402      BEQ      5$      ;;BR IF NONE
1095 002746 013716 001206      MOV      $ESCAPE,(SP)  ;;FUDGE RETURN ADDRESS FOR ESCAPE
1096 002752      5$:
1097 002752 032737 001000 177570      BIT      #SW9,SWR      ;;ARE YOU LOOPING ON THIS ERROR?
1098 002760 001425      BEQ      EEXIT      ;;BRANCH IF NOT LOOPING ON ERROR
1099 002762 012737 177777 177766      MOV      #-1,@#CPUERR  ;;CLEAR CPU ERROR REGISTER
1100 002770 012737 177777 177744      MOV      #-1,@#MEMERR  ;;CLEAR MEMORY ERROR REGISTER
1101 002776 042737 177776 177572      BIC      #177776,@#MMRO  ;;CLEAR MEMORY MANAGEMENT STATUS REGISTER
1102 003004 012737 177777 005154      MOV      #-1,TOFLAG  ;;INITIALIZE TRAP FLAG
1103 003012 012737 177777 005654      MOV      #-1,CPFLAG  ;;INITIALIZE CP TRAP FLAG
1104 003020 012737 177777 005766      MOV      #-1,PAFLAG  ;;INITIALIZE PARITY TRAP FLAG
1105 003026 012737 177777 006212      MOV      #-1,MMFLAG  ;;INITIALIZE MEMORY MANAGEMENT TRAP FLAG
1106 003034 000002      EEXIT: RTI      ;;RETURN TO TEST
```

.SBTTL ERROR MESSAGE TYPE OUT ROUTINE

```
1107
1108
1109
1110
1111
1112      ;*      THIS SUBROUTINE IS CALLED BY THE ERROR HANDLER TO TYPE
1113      ;*      THE ERROR MESSAGES. IT PICKS UP THE ITEM BYTE ($ITEMB) NUMBER
1114      ;*      AND USES THAT TO INDEX THROUGH THE ERROR TABLE. THE ERROR
1115      ;*      TABLE STARTS AT '$ERRTB' AND HAS FOUR (4) POINTERS FOR EACH
1116      ;*      ENTRY, 'EM', 'DH', 'DT', 'DF'. THE 'EM' POINTS TO THE ERROR
1117      ;*      MESSAGE WHICH IS AN ASCII STRING. THE 'DH' POINTS TO THE DATA
1118      ;*      HEADER WHICH IS ANOTHER ASCII STRING. THE 'DT' POINTS TO THE
1119      ;*      DATA TABLE WHICH IS A GROUP OF WORDS CONTAINING THE ADDRESSES
1120      ;*      OF THE DATA TO BE TYPED. THE FORMAT OF THIS DATA IS
1121      ;*      CONTROLLED BY THE 'DF' WHICH IS THE POINTER TO THE DATA FORMAT.
1122      ;*      THE DATA FORMAT IS A GROUP OF BYTES WHICH CONTAIN NUMBERS
1123      ;*      THAT CORRESPOND TO DIFFERENT TYPING FORMATS.
1124      ;*      0      -16 BIT OCTAL FORMAT
1125      ;*      1      -DECIMAL FORMAT
```

```
1126      ;*      2      -22 BIT OCTAL FORMAT. DATA IS LOWER 16 BITS OF THE
1127      ;*      ;*      PHYSICAL ADDRESS, UPPER 6 BITS ARE ADJACENT TO LOWER 16
1128      ;*      3      -22 BIT OCTAL FORMAT. DATA IS THE 16 BIT VIRTUAL
1129      ;*      ;*      ADDRESS IN KERNEL I-SPACE.
1130      ;*      4      -18 BIT OCTAL FORMAT. DATA IS A 16 BIT NUMBER THAT
1131      ;*      ;*      WILL BE CONVERTED INTO A UNIBUS ADDRESS BY LEFT
1132      ;*      ;*      SHIFTING IT 6 BITS.
1133 003036 010046      ERTYPE: MOV      R0,-(KSP)      ;SAVE R0 ON STACK
1134 003040 005000      CLR      R0      ;CLEAR R0
1135 003042 113700 001114      MOV      @#ITEMB,R0      ;PUT ITEM NUMBER IN R0
1136 003046 001004      BNE      1$      ;BRANCH IF IT IS NON-ZERO
1137 003050 013746 001116      MOV      $ERRPC,-(KSP)      ;PUT ERROR PC ON STACK FOR TYPING
1138 003054 104402      TYPOC      ;TYPE FAILING PC
1139 003056 000526      BR      13$      ;GO TO RETURN
1140 003060 005300      1$: DEC      R0      ;ADJUST ITEM NUMBER TO BE A POINTER
1141 003062 072027 000003      ASH      #3,R0      ;LEFT SHIFT ITEM NO. 3 PLACES
1142 003066 100024      BPL      22$      ;BRANCH IF ITEM # LESS THAN 200
1143 003070 032700 001000      BIT      #BIT9,R0      ;SEE IF ITEM # WAS 3XX
1144 003074 001415      BEQ      21$      ;BRANCH IF ITEM # WAS 2XX
1145      ;AND TYPE ERROR MESSAGE
1146 003076 032737 000200 177570      BIT      #SW7,@#SWR      ;SEE IF SWITCH 7 IS UP
1147 003104 001404      BEQ      20$      ;BRANCH IF SWITCH IS NOT UP
1148      ;AND TYPE DATA, ON MULTIPLE ERRORS
1149 003106 062766 000004 000002      ADD      #4,2(KSP)      ;SKIP 'TYPE', $CRLF' IF SW 7 IS UP
1150      ;INHIBIT MULTIPLE ERROR TYPEOUTS
1151 003114 000507      BR      13$      ;BRANCH TO EXIT
1152 003116 042700 177000      20$: BIC      #177000,R0      ;CLEAR UPPER BYTE OF R0
1153 003122 062700 002130      ADD      #ER200+4,R0      ;POINT TO DATA TABLE ENTRY
1154 003126 000424      BR      5$      ;GO TYPE DATA TABLE
1155 003130 042700 177000      21$: BIC      #177000,R0      ;CLEAR UPPER BYTE OF R0
1156 003134 062700 000570      ADD      #<ER200-$ERRTB>,R0 ;ADD DIFFERENCE BETWEEN
1157      ;ITEM 1 AND ITEM 201
1158      ;GET POINTER TO ERROR MESSAGE AND TYPE IT
1159      ;IF THE POINTER IS NOT ZERO
1160 003140 062700 001334      22$: ADD      #$ERRTB,R0      ;ADD BASE OF ERROR TABLE
1161 003144 012037 003154      MOV      (R0)+,2$      ;PUT MESSAGE POINTER IN TYPE STATEMENT
1162 003150 001404      BEQ      3$      ;BRANCH IF NO ERROR MESSAGE
1163 003152 104400      TYPE      ;TYPE ERROR MESSAGE
1164 003154 000000      2$: .WORD      0      ;POINTER TO ERROR MESSAGE
1165 003156 104400 001215      TYPE      ,$CRLF      ;TYPE CRLF
1166      ;GET THE POINTER TO THE DATA HEADER AND
1167      ;TYPE IT IF THE POINTER IS NOT ZERO
1168 003162 012037 003172      3$: MOV      (R0)+,4$      ;PUT HEADER POINTER IN TYPE STATEMENT
1169 003166 001404      BEQ      5$      ;BRANCH IF NO DATA HEADER
1170 003170 104400      TYPE      ;TYPE THE DATA HEADER
1171 003172 000000      4$: .WORD      0      ;POINTER TO DATA HEADER
1172 003174 104400 001215      TYPE      ,$CRLF      ;TYPE CRLF
1173      ;THIS IS THE START OF THE DATA OUTPUT IF THE
1174      ;DATA POINTER IS NOT ZERO. R0 POINTS TO THE
1175      ;DATA FORMAT, R1 POINTS TO THE ADDRESS OF
1176      ;THE DATA WORDS.
1177 003200 010146      5$: MOV      R1,-(KSP)      ;SAVE R1 ON THE STACK
1178 003202 012001      MOV      (R0)+,R1      ;PUT DATA TABLE POINTER IN R1
1179 003204 001452      BEQ      12$      ;BRANCH IF NO DATA TABLE
1180 003206 012000      MOV      (R0)+,R0      ;PICK UP DATA FORMAT POINTER
1181 003210 105710      6$: TSTB      (R0)      ;IS THIS WORD OCTAL
```



```

1182 003212 001003      BNE      7$      ;BRANCH IF NOT 16-BIT OCTAL
1183      ::      THIS IS 16 BIT OCTAL FORMAT (DF = 0)
1184 003214 013146      MOV      @ (R1)+, -(KSP) ;PUT WORD ON STACK FOR TYPING
1185 003216 104402      TYPDC      ;TYPE THE WORD ON STACK AS 16 BIT OCTAL
1186 003220 000436      BR      11$      ;GET READY FOR NEXT WORD
1187 003222 122710 000001 7$:      CMPB     #1, (R0) ;IS THE WORD DECIMAL
1188 003226 001003      BNE      8$      ;BRANCH IF NOT DECIMAL
1189      ::      THIS IS DECIMAL FORMAT (DF = 1)
1190 003230 013146      MOV      @ (R1)+, -(KSP) ;PUT WORD ON STACK FOR TYPING
1191 003232 104410      TYPDC      ;TYPE THE WORD ON STACK AS DECIMAL
1192 003234 000430      BR      11$      ;GET READY FOR NEXT WORD
1193 003236 122710 000002 8$:      CMPB     #2, (R0) ;IS WORD 22-BIT PHYSICAL ADDRESS
1194 003242 001012      BNE      9$      ;BRANCH IF NOT 22-BIT PHYSICAL ADDR
1195      ::      THIS IS 22-BIT PHYSICAL FORMAT (DF = 2)
1196 003244 012146      MOV      (R1)+, -(KSP) ;PUT ADDR OF LOW WORD ON STACK
1197 003246 004737 004750 JSR      PC, $DB20 ;CONVERT NUMBER TO OCTAL ASCIZ
1198 003252 062716 000003 ADD      #3, (KSP) ;ONLY WANT 8 DIGITS
1199 003256 012637 003264 MOV      (KSP)+, 30$ ;PUT POINTER AFTER 'TYPE' CALL
1200 003262 104400      TYPE      ;TYPE ASCIZ STRING
1201 003264 000000 30$:      .WORD     0 ;WORD HOLDS POINTER TO ASCIZ STRING
1202 003266 000413      BR      11$      ;GET READY FOR NEXT WORD
1203 003270 122710 000003 9$:      CMPB     #3, (R0) ;IS THIS A 16-BIT VIRTUAL ADDRESS
1204 003274 001004      BNE      10$     ;BRANCH IF NOT 16-BIT VIRT. ADDR.
1205      ::      THIS IS 22-BIT VIRTUAL ADDRESS FORMAT
1206      ::      KERNEL I-SPACE ASSUMED. (DF = 3)
1207 003276 013146      MOV      @ (R1)+, -(KSP) ;PUT 16-BIT VIRTUAL ADDR ON STACK
1208 003300 004737 003344 JSR      PC, TYPVAD ;GO TYPE 22-BIT ADDRESS FROM 16-BIT V.A.
1209 003304 000404      BR      11$      ;GET READY FOR NEXT WORD
1210      ::      THIS IS FORMAT 4. DATA WORD IS A UNIBUS ADDRESS
1211      ::      OUTPUT WILL BE 18-BITS WORD LEFT SHIFTED 6.
1212 003306 013146 10$:      MOV      @ (R1)+, -(KSP) ;PUSH 16-BIT UNIBUS ADDRESS ON STACK
1213 003310 004737 003452 JSR      PC, UBADDR ;CONVERT TO 18-BIT UNIBUS ADDR AND TYPE
1214 003314 000400      BR      11$      ;GET READY FOR NEXT WORD
1215 003316 005200 11$:      INC      R0 ;POINT TO NEXT FORMAT BYTE
1216 003320 104400 003340 TYPE      ,32$ ;TYPE TWO SPACES
1217 003324 005711      TST      (R1) ;IS THERE ANOTHER WORD?
1218 003326 001401      BEQ      12$      ;BRANCH IF ALL DONE
1219 003330 000727      BR      6$      ;GO BACK FOR NEXT NUMBER
1220 003332 012601 12$:      MOV      (KSP)+, R1 ;RESTORE R1
1221 003334 012600 13$:      MOV      (KSP)+, R0 ;RESTORE R0
1222 003336 000207      RTS      PC ;RETURN TO ERROR ROUTINE
1223 003340 020040 000 32$:      .ASCIZ     ? ? ;TWO SPACES
1224      .EVEN
1225
1226
1227      .SBTTL  CONVERT 16-BIT VIRTUAL ADDRESS TO 22-BIT PHYSICAL ADDRESS
1228      ;*
1229      ;* THIS ROUTINE IS CALLED BY A 'JSR PC' AFTER THE VIRTUAL ADDRESS
1230      ;* IS PUSHED ON THE KERNEL STACK. THE V.A. IS THEN LOADED INTO
1231      ;* R1 AND THE UPPER 3 BITS ARE SHIFTED INTO R0 TO SELECT THE
1232      ;* CORRECT KERNEL I-SPACE PAR. THE LOWER 12 BITS OF THE VIRTUAL
1233      ;* ADDRESS ARE ADDED TO THE PAR AS THEY ARE BY MEMORY MANAGEMENT
1234      ;* AND THE PHYSICAL ADDRESS IS SAVED IN MEMORY TO BE CONVERTED
1235      ;* TO ASCIZ AND TYPED.
1236      ;*
1237 003344 104412      TYPVAD: SAVREG ;SAVE ALL REGISTERS

```

1238	003346	016601	000002	MOV	2(KSP),R1	:PUT VIRTUAL ADDR IN R1
1239	003352	005000		CLR	R0	:CLEAR R0 FOR CALCULATIONS
1240	003354	073027	000003	ASHC	#3,R0	:LEFT SHIFT R0,R1 3 PLACES
1241	003360	006300		ASL	R0	:LEFT SHIFT R0 ONE MORE PLACE
1242	003362	006001		ROR	R1	:RIGHT SHIFT R1 SO OFFSET IS CORRECT
1243	003364	006001		ROR	R1	:RIGHT SHIFT R1
1244	003366	006001		ROR	R1	:RIGHT SHIFT R1
1245	003370	062700	172340	ADD	#KIPAR0,R0	:FORM DESIRED PAR ADDR IN R0
1246	003374	011003		MOV	(R0),R3	:PUT CONTENTS OF PAR IN R3
1247	003376	005002		CLR	R2	:CLEAR R2 FOR PHYSICAL ADDR CALCULATIONS
1248	003400	073227	000006	ASHC	#6,R2	:LEFT SHIFT <R2,R3> 6 PLACES
1249	003404	060103		ADD	R1,R3	:ADD OFFSET IN R1 TO BASE IN R3
1250	003406	005502		ADC	R2	:ADD ANY POSSIBLE CARRY TO UPPER 6 BITS
1251	003410	010237	001222	MOV	R2,PADRSR	:PUT UPPER 6 BITS OF ADDR IN CORE
1252	003414	010337	001220	MOV	R3,PADRSL	:PUT LOWER 16 BITS OF ADDR IN CORE
1253	003420	012746	001220	MOV	#PADRSL,-(KSP)	:PUT POINTER TO LOWER 16 BITS ON STACK
1254	003424	004737	004750	JSR	PC,\$DB20	:CONVERT NUMBER TO OCTAL ASCII
1255	003430	062716	000003	ADD	#3,(KSP)	:ONLY TYPE 8 DIGITS
1256	003434	012637	003442	MOV	(KSP)+,3\$	:PUT POINTER AFTER TYPE INST
1257	003440	104400		TYPE		:TYPE THE 22-BIT VIRTUAL ADDRESS
1258	003442	000000		3\$: .WORD	0	:THIS WORD HOLDS THE POINTER TO
1259						:THE ASCII STRING
1260	003444	104414		RESREG		:RESTORE ALL THE REGISTERS
1261	003446	012616		MOV	(KSP)+,(KSP)	:LEAVE ONLY RETURN ADDR ON STACK
1262	003450	000207		RTS	PC	:RETURN TO ERROR HANDLER

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

1263				UBADDR: MOV	2(KSP),R1	:LOAD 16 BIT ADDRESS INTO R1
1264				CLR	R0	:CLEAR R0 FOR CALCULATIONS
1265				ASHC	#6,R0	:LEFT SHIFT <R0,R1> 6 PLACES
1266				MOV	R1,PADRSL	:PUT LOWER 16 BITS IN PADRSR
1267				MOV	R0,PADRSR	:PUT UPPER 6 BITS IN PADRSR
1268				MOV	#PADRSR,-(KSP)	:PUSH POINTER TO WORDS ON STACK
1269	003452	016601	000002	JSR	PC,\$DB20	:JUMP TO CONVERT ROUTINE
1270	003456	005000		ADD	#5,(KSP)	:ONLY USE LOWER 6 CHARS.
1271	003460	073027	000006	MOV	(KSP)+,3\$	:PUT POINTER AFTER TYPE CALL.
1272	003464	010137	001220	TYPE		
1273	003470	010037	001222	3\$: .WORD	0	:HOLDS POINTER TO FIRST CHAR.
1274	003474	012746	001220	MOV	(KSP)+,(KSP)	:LEAVE ONLY RETURN ADDRESS ON STACK
1275	003500	004737	004750	RTS	PC	:RETURN TO ERROR TYPE ROUTINE.
1276	003504	062716	000005			
1277	003510	012637	003516			
1278	003514	104400				
1279	003516	000000				
1280	003520	012616				
1281	003522	000207				

::*****

.SBTTL SAVE AND RESTORE R0-R5 ROUTINES

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


```

1294      ; * +2---(+18)
1295      ; * +4---R5
1296      ; * +6---R4
1297      ; * +8---R3
1298      ; * +10---R2
1299      ; * +12---R1
1300      ; * +14---R0
1301
1302      $SAVREG:
1303      MOV      R0,-(SP)      ;; PUSH R0 ON STACK
1304      MOV      R1,-(SP)      ;; PUSH R1 ON STACK
1305      MOV      R2,-(SP)      ;; PUSH R2 ON STACK
1306      MOV      R3,-(SP)      ;; PUSH R3 ON STACK
1307      MOV      R4,-(SP)      ;; PUSH R4 ON STACK
1308      MOV      R5,-(SP)      ;; PUSH R5 ON STACK
1309      MOV      22(SP),-(SP)   ;; SAVE PS OF MAIN FLOW
1310      MOV      22(SP),-(SP)   ;; SAVE PC OF MAIN FLOW
1311      MOV      22(SP),-(SP)   ;; SAVE PS OF CALL
1312      MOV      22(SP),-(SP)   ;; SAVE PC OF CALL
1313      RTI
1314
1315      ; *RESTORE R0-R5
1316      ; *CALL:
1317      ; *      RESREG
1318      $RESREG:
1319      MOV      (SP)+,22(SP)   ;; RESTORE PC OF CALL
1320      MOV      (SP)+,22(SP)   ;; RESTORE PS OF CALL
1321      MOV      (SP)+,22(SP)   ;; RESTORE PC OF MAIN FLOW
1322      MOV      (SP)+,22(SP)   ;; RESTORE PS OF MAIN FLOW
1323      MOV      (SP)+,R5
1324      MOV      (SP)+,R4
1325      MOV      (SP)+,R3
1326      MOV      (SP)+,R2
1327      MOV      (SP)+,R1
1328      MOV      (SP)+,R0
1329      RTI
1330      ;; *****
1331
1332      .SBTTL  TYPE ROUTINE
1333
1334      ; *ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
1335      ; *THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
1336      ; *NOTE1:      $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
1337      ; *NOTE2:      $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
1338      ; *NOTE3:      $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
1339      ; *
1340      ; *CALL:
1341      ; *1) USING A TRAP INSTRUCTION
1342      ; *      TYPE      ,MESADR      ;; MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
1343      ; *OR
1344      ; *      TYPE
1345      ; *      MESADR
1346      ; *
1347      ; *2) USING A JSR INSTRUCTION
1348      ; *      MOV      PS,-(SP)      ;; PUSH PROCESSOR STATUS WORD ON THE STACK
1349      ; *      JSR      PC,$TYPE      ;; CALL TYPE ROUTINE

```

```
1350      ;*      MESADDR      ;;FIRST ADRESS OF MESSAGE
1351
1352 003620 105737 001151 $TYPE: TSTB $TPFLG      ;;IS THERE A TERMINAL?
1353 003624 100002      BPL 1$      ;;BR IF YES
1354 003626 000000      HALT      ;;HALT HERE IF NO TERMINAL
1355 003630 000407      BR 3$      ;;LEAVE
1356 003632 010046 1$: MOV R0,-(SP)      ;;SAVE R0
1357 003634 017600 000002      MOV @2(SP),R0      ;;GET ADDRESS OF ASCII STRING
1358 003640 112046      MOVB (R0)+,-(SP)      ;;PUSH CHARACTER TO BE TYPED ONTO STACK
1359 003642 001005      BNE 4$      ;;BR IF IT ISN'T THE TERMINATOR
1360 003644 005726      TST (SP)+      ;;IF TERMINATOR POP IT OFF THE STACK
1361 003646 012600      MOV (SP)+,R0      ;;RESTORE R0
1362 003650 062716 000002 3$: ADD #2,(SP)      ;;ADJUST RETURN PC
1363 003654 000002      RTI      ;;RETURN
1364 003656 122716 000011 4$: CMPB #HT,(SP)      ;;BRANCH IF <HT>
1365 003662 001426      BEQ 8$
1366 003664 122716 000200      CMPB #CRLF,(SP)      ;;BRANCH IF NOT
1367 003670 001004      BNE 5$
1368 003672 005726      TST (SP)+      ;;POP <CR><LF> EQUIV
1369 003674 104400 001215      TYPE $CRLF
1370 003700 000757      BR 2$      ;;GET NEXT CHARACTER
1371 003702 004737 003764 5$: JSR PC,$TYPEPC      ;;GO TYPE THIS CHARACTER
1372 003706 123726 001150 6$: CMPB $FILLC,(SP)+      ;;IS IT TIME FOR FILLER CHARS.?
1373 003712 001352      BNE 2$      ;;IF NO GO GET NEXT CHAR.
1374 003714 013746 001146      MOV $NULL,-(SP)      ;;GET # OF FILLER CHARS. NEEDED
1375      ;;AND THE NULL CHAR.
1376 003720 105366 000001 7$: DECB 1(SP)      ;;DOES A NULL NEED TO BE TYPED?
1377 003724 002770      BLT 6$      ;;BR IF NO--GO POP THE NULL OFF OF STACK
1378 003726 004737 003764      JSR PC,$TYPEPC      ;;GO TYPE A NULL
1379 003732 105337 004030      DECB $CHARCNT      ;;DON'T COUNT THE NULL AS A CHARACTER
1380 003736 000770      BR 7$      ;;LOOP
1381
1382      ;;HORIZONTAL TAB PROCESSOR
1383
1384 003740 112716 000040 8$: MOVB #' ,(SP)      ;;REPLACE TAB WITH SPACE
1385 003744 004737 003764 9$: JSR PC,$TYPEPC      ;;TYPE A SPACE
1386 003750 132737 000007 004030      BITB #7,$CHARCNT      ;;BRANCH IF NOT AT
1387 003756 001372      BNE 9$      ;;TAB STOP
1388 003760 005726      TST (SP)+      ;;POP SPACE OFF STACK
1389 003762 000726      BR 2$      ;;GET NEXT CHARACTER
1390 003764 105777 175152 $TYPEPC: TSTB @2$TPS      ;;WAIT UNTIL PRINTER IS READY
1391 003770 100375      BPL $TYPEPC
1392 003772 116677 000002 175144      MOVB 2(SP),@2$TPB      ;;LOAD CHAR TO BE TYPED INTO DATA REG.
1393 004000 122766 000015 000002      CMPB #CR,2(SP)      ;;BRANCH IF
1394 004006 001003      BNE 1$      ;;NOT <CR>
1395 004010 105037 004030      CLRB $CHARCNT      ;;
1396 004014 000406      BR $TYPEPC      ;;EXIT
1397 004016 122766 000012 000002 1$: CMPB #LF,2(SP)      ;;BRANCH IF
1398 004024 001402      BEQ $TYPEPC      ;;<LF>
1399 004026 105227      INCB (PC)+      ;;INC SPACE
1400 004030 000000      $CHARCNT: .WORD 0      ;;COUNT
1401 004032 000207      $TYPEPC: RTS PC
1402
1403      ;;*****
1404      .SBTTL BINARY TO OCTAL (ASCII) AND TYPE
1405
```



```
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429 004034 017646 000000
1430 004040 116637 000001 004257
1431 004046 112637 004261
1432 004052 062716 000002
1433 004056 000406
1434 004060 112737 000001 004257
1435 004066 112737 000006 004261
1436 004074 112737 000005 004256
1437 004102 010346
1438 004104 010446
1439 004106 010546
1440 004110 113704 004261
1441 004114 005404
1442 004116 062704 000006
1443 004122 110437 004260
1444 004126 113704 004257
1445 004132 016605 000012
1446 004136 005003
1447 004140 006105
1448 004142 000404
1449 004144 006105
1450 004146 006105
1451 004150 006105
1452 004152 010503
1453 004154 006103
1454 004156 105337 004260
1455 004162 100016
1456 004164 042703 177770
1457 004170 001002
1458 004172 005704
1459 004174 001403
1460 004176 005204
1461 004200 052703 000060

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

1462 004204 052703 000040
1463 004210 110337 004254
1464 004214 104400 004254
1465 004220 105337 004256
1466 004224 003347
1467 004226 002402
1468 004230 005204
1469 004232 000744
1470 004234 012605
1471 004236 012604
1472 004240 012603
1473 004242 016666 000002 000004
1474 004250 012616
1475 004252 000002
1476 004254 000
1477 004255 000
1478 004256 000
1479 004257 000
1480 004260 000000
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494 004262
1495 004262 010046
1496 004264 010146
1497 004266 010246
1498 004270 010346
1499 004272 010546
1500 004274 012746 020200
1501 004300 016605 000020
1502 004304 100004
1503 004306 005405
1504 004310 112766 000055 000001
1505 004316 005000
1506 004320 012703 004476
1507 004324 112723 000040
1508 004330 005002
1509 004332 016001 004466
1510 004336 160105
1511 004340 002402
1512 004342 005202
1513 004344 000774
1514 004346 060105
1515 004350 005702
1516 004352 001002
1517 004354 105716

```

5$:  BIS    #' ,R3      ;;MAKE ASCII IF NOT ALREADY
      MOVB   R3 ,8$      ;;SAVE FOR TYPING
      TYPE   ,8$         ;;GO TYPE THIS DIGIT
7$:  DECB    $OCNT      ;;COUNT BY 1
      BGT     2$         ;;BR IF MORE TO DO
      BLT     6$         ;;BR IF DONE
      INC     R4         ;;INSURE LAST DIGIT ISN'T A BLANK
      BR      2$         ;;GO DO THE LAST DIGIT
6$:  MOV     (SP)+ ,R5    ;;RESTORE R5
      MOV     (SP)+ ,R4    ;;RESTORE R4
      MOV     (SP)+ ,R3    ;;RESTORE R3
      MOV     2(SP) ,4(SP) ;;SET THE STACK FOR RETURNING
      MOV     (SP)+ , (SP)
      RTI              ;;RETURN
8$:  .BYTE    0          ;;STORAGE FOR ASCII DIGIT
      .BYTE    0          ;;TERMINATOR FOR TYPE ROUTINE
$OCNT: .BYTE    0         ;;OCTAL DIGIT COUNTER
$OFILL: .BYTE    0        ;;ZERO FILL SWITCH
$OMODE: .WORD    0        ;;NUMBER OF DIGITS TO TYPE
;*****
.SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
;THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
;SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
;NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
;BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
;REPLACED WITH SPACES.
;CALL:
;*      MOV     NUM,-(SP)    ;;PUT THE BINARY NUMBER ON THE STACK
;*      TYPDS          ;;GO TO THE ROUTINE
$TYPDS:
      MOV     R0,-(SP)      ;;PUSH R0 ON STACK
      MOV     R1,-(SP)      ;;PUSH R1 ON STACK
      MOV     R2,-(SP)      ;;PUSH R2 ON STACK
      MOV     R3,-(SP)      ;;PUSH R3 ON STACK
      MOV     R5,-(SP)      ;;PUSH R5 ON STACK
      MOV     #20200,-(SP)  ;;SET BLANK SWITCH AND SIGN
      MOV     20(SP) ,R5    ;;GET THE INPUT NUMBER
      BPL     1$           ;;BR IF INPUT IS POS.
      NEG     R5           ;;MAKE THE BINARY NUMBER POS.
      MOVB    #'-,1(SP)    ;;MAKE THE ASCII NUMBER NEG.
1$:  CLR     R0            ;;ZERO THE CONSTANTS INDEX
      MOV     #$DBLK ,R3    ;;SETUP THE OUTPUT POINTER
      MOVB    #' , (R3)+   ;;SET THE FIRST CHARACTER TO A BLANK
2$:  CLR     R2            ;;CLEAR THE BCD NUMBER
      MOV     $DTBL(R0) ,R1 ;;GET THE CONSTANT
3$:  SUB     R1 ,R5        ;;FORM THIS BCD DIGIT
      BLT     4$           ;;BR IF DONE
      INC     R2           ;;INCREASE THE BCD DIGIT BY 1
      BR      3$
4$:  ADD     R1 ,R5        ;;ADD BACK THE CONSTANT
      TST     R2           ;;CHECK IF BCD DIGIT=0
      BNE     5$          ;;FALL THROUGH IF 0
      TSTB    (SP)        ;;STILL DOING LEADING 0'S?

```



```
1518 004356 100407
1519 004360 106316
1520 004362 103003
1521 004364 116663 000001 177777
1522 004372 052702 000060
1523 004376 052702 000040
1524 004402 110223
1525 004404 005720
1526 004406 020027 000010
1527 004412 002746
1528 004414 003002
1529 004416 010502
1530 004420 000764
1531 004422 105726
1532 004424 100003
1533 004426 116663 177777 177776
1534 004434 105013
1535 004436 012605
1536 004440 012603
1537 004442 012602
1538 004444 012601
1539 004446 012600
1540 004450 104400 004476
1541 004454 016666 000002 000004
1542 004462 012616
1543 004464 000002
1544 004466 023420
1545 004470 001750
1546 004472 000144
1547 004474 000012
1548 004476 000004
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558 004506 010046
1559 004510 016600 000002
1560 004514 005740
1561 004516 111000
1562 004520 016000 004526
1563 004524 000200
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573 004526
```

```

      BMI      7$          ;;BR IF YES
5$:    ASLB     (SP)        ;;MSD?
      BCC      6$          ;;BR IF NO
      MOVB     1(SP),-1(R3) ;;YES--SET THE SIGN
6$:    BIS      #'0,R2     ;;MAKE THE BCD DIGIT ASCII
7$:    BIS      #' ,R2     ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
      MOVB     R2,(R3)+    ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
      TST      (R0)+       ;;JUST INCREMENTING
      CMP      R0,#10      ;;CHECK THE TABLE INDEX
      BLT      2$          ;;GO DO THE NEXT DIGIT
      BGT      8$          ;;GO TO EXIT
      MOV      R5,R2       ;;GET THE LSD
      BR       6$          ;;GO CHANGE TO ASCII
8$:    TSTB     (SP)+       ;;WAS THE LSD THE FIRST NON-ZERO?
      BPL      9$          ;;BR IF NO
9$:    MOVB     -1(SP),-2(R3) ;;YES--SET THE SIGN FOR TYPING
      CLRB     (R3)        ;;SET THE TERMINATOR
      MOV      (SP)+,R5     ;;POP STACK INTO R5
      MOV      (SP)+,R3     ;;POP STACK INTO R3
      MOV      (SP)+,R2     ;;POP STACK INTO R2
      MOV      (SP)+,R1     ;;POP STACK INTO R1
      MOV      (SP)+,R0     ;;POP STACK INTO R0
      TYPE     $DBLK        ;;NOW TYPE THE NUMBER
      MOV      2(SP),4(SP) ;;ADJUST THE STACK
      MOV      (SP)+,(SP)
      RTI              ;;RETURN TO USER

$DTBL: 10000.
      1000.
      100.
      10.
$DBLK: .BLKW 4
;*****
.SBTTL TRAP DECODER

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

$TRAP: MOV      R0,-(SP)    ;;SAVE R0
      MOV      2(SP),R0    ;;GET TRAP ADDRESS
      TST      -(R0)       ;;BACKUP BY 2
      MOVB     (R0),R0     ;;GET RIGHT BYTE OF TRAP
      MOV      $TRPAD(R0),R0 ;;INDEX TO TABLE
      RTS      R0          ;;GO TO ROUTINE

.SBTTL TRAP TABLE

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

:      ROUTINE
:      -----
$TRPAD:
```

1574	004526	003620	\$TYPE	::CALL=TYPE	TRAP+0(104400)	TTY TYPEOUT ROUTINE
1575	004530	004060	\$TYPOC	::CALL=TYPOC	TRAP+2(104402)	TYPE OCTAL NUMBER (WITH LEADING ZEROS)
1576	004532	004034	\$TYPOS	::CALL=TYPOS	TRAP+4(104404)	TYPE OCTAL NUMBER (NO LEADING ZEROS)
1577	004534	004074	\$TYPON	::CALL=TYPON	TRAP+6(104406)	TYPE OCTAL NUMBER (AS PER LAST CALL)
1578	004536	004262	\$TYPDS	::CALL=TYPDS	TRAP+10(104410)	TYPE DECIMAL NUMBER (WITH SIGN)
1579	004540	003524	\$SAVREG	::CALL=SAVREG	TRAP+12(104412)	SAVE R0-R5 ROUTINE
1580	004542	003562	\$RESREG	::CALL=RESREG	TRAP+14(104414)	RESTORE R0-R5 ROUTINE
1581	004544	005070	TBITOFF	::CALL=TBITO	TRAP+16(104416)	THIS WILL TURN OFF T BIT TRAPPING
1582	004546	005116	TBITRESTORE	::CALL=TBITR	TRAP+20(104420)	THIS WILL RETURN THE T BIT TO PR

::*****

.SBTTL POWER DOWN AND UP ROUTINES

:POWER DOWN ROUTINE

1589	004550	012737	004676	000024	\$PWRDN: MOV	#\$ILLUP, @#PWRVEC	::SET FOR FAST UP
1590	004556	012737	000340	000026	MOV	#340, @#PWRVEC+2	::PRIO:7
1591	004564	010046			MOV	R0, -(SP)	::PUSH R0 ON STACK
1592	004566	010146			MOV	R1, -(SP)	::PUSH R1 ON STACK
1593	004570	010246			MOV	R2, -(SP)	::PUSH R2 ON STACK
1594	004572	010346			MOV	R3, -(SP)	::PUSH R3 ON STACK
1595	004574	010446			MOV	R4, -(SP)	::PUSH R4 ON STACK
1596	004576	010546			MOV	R5, -(SP)	::PUSH R5 ON STACK
1597	004600	010637	004702		MOV	SP, \$SAVR6	::SAVE SP
1598	004604	012737	004616	000024	MOV	#\$PWRUP, @#PWRVEC	::SET UP VECTOR
1599	004612	000000			HALT		
1600	004614	000776			BR	.-2	::HANG UP

:POWER UP ROUTINE

1603	004616	013706	004702		\$PWRUP: MOV	\$SAVR6, SP	::GET SP
1604	004622	005037	004702		CLR	\$SAVR6	::WAIT LOOP FOR THE TTY
1605	004626	005237	004702		1\$: INC	\$SAVR6	::WAIT FOR THE INC
1606	004632	001375			BNE	1\$	::OF WORD
1607	004634	012605			MOV	(SP)+, R5	::POP STACK INTO R5
1608	004636	012604			MOV	(SP)+, R4	::POP STACK INTO R4
1609	004640	012603			MOV	(SP)+, R3	::POP STACK INTO R3
1610	004642	012602			MOV	(SP)+, R2	::POP STACK INTO R2
1611	004644	012601			MOV	(SP)+, R1	::POP STACK INTO R1
1612	004646	012600			MOV	(SP)+, R0	::POP STACK INTO R0
1613	004650	012737	004550	000024	MOV	#\$PWRDN, @#PWRVEC	::SET UP THE POWER DOWN VECTOR
1614	004656	012737	000340	000026	MOV	#340, @#PWRVEC+2	::PRIO:7
1615	004664	104400			TYPE		::REPORT THE POWER FAILURE
1616	004666	004704			\$PWRMG: .WORD	PWRMSG	::POWER FAIL MESSAGE POINTER
1617	004670	012716			MOV	(PC)+, (SP)	::RESTART AT START
1618	004672	010000			\$PWRAD: .WORD	START	::RESTART ADDRESS
1619	004674	000002			RTI		
1620	004676	000000			\$ILLUP: HALT		::THE POWER UP SEQUENCE WAS STARTED
1621	004700	000776			BR	.-2	::BEFORE THE POWER DOWN WAS COMPLETE
1622	004702	000000			\$SAVR6: 0		::PUT THE SP HERE
1623	004704	006412	047520	042527	PWRMSG: .ASCIZ	<12><15>?POWER FAILURE, RESTARTING PROGRAM?	
1624	004712	020122	040506	046111			
1625	004720	051125	026105	051040			
1626	004726	051505	040524	052122			
1627	004734	047111	020107	051120			
1628	004742	043517	040522	000115			
1629					.EVEN		


```
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643 004750 104412
1644 004752 016601 000002
1645 004756 012705 005067
1646 004762 012704 000014
1647 004766 012703 177770
1648 004772 012100
1649 004774 012101
1650 004776 005002
1651 005000 110245
1652 005002 010002
1653 005004 005304
1654 005006 003007
1655 005010 001405
1656 005012 005205
1657 005014 010566 000002
1658 005020 104414
1659 005022 000207
1660 005024 006203
1661 005026 006001
1662 005030 006000
1663 005032 006001
1664 005034 006000
1665 005036 006001
1666 005040 006000
1667 005042 040302
1668 005044 062702 000060
1669 005050 000753
1670 005052 000016
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
```

```
::*****
.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

$DB20: SAVREG                      ;; SAVE ALL REGISTERS
      MOV      2(SP), R1           ;; PICKUP THE POINTER TO LOW WORD
      MOV      #SOCTVL+13., R5    ;; POINTER TO DATA TABLE
      MOV      #12., R4           ;; DO ELEVEN CHARACTERS
      MOV      #^C7, R3          ;; MASK
      MOV      (R1)+, R0          ;; LOWER WORD
      MOV      (R1)+, R1          ;; HIGH WORD
      CLR      R2                 ;; TERMINATOR
1$:     MOV      R2, -(R5)          ;; PUT CHARACTER IN DATA TABLE
      MOV      R0, R2             ;; GET THIS DIGIT
      DEC      R4                 ;; COUNT THIS CHARACTER
      BGT      3$                 ;; 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
2$:     ASR      R3                 ;; POSITION THE MASK FOR THE LAST DIGIT
3$:     ROR      R1                 ;; POSITION THE BINARY NUMBER FOR
      ROR      R0                 ;; THE NEXT OCTAL DIGIT
      ROR      R1
      ROR      R0
      ROR      R1
      ROR      R0
      BIC      R3, R2              ;; MASK OUT ALL JUNK
      ADD      #'0, R2            ;; MAKE THIS CHAR. ASCII
      BR       1$                 ;; GO PUT IT IN THE DATA TABLE
$SOCTVL: .BLKB 14.                ;; RESERVE DATA TABLE

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

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

```
1686
1687 005070
1688 005070 032766 000020 000002
1689 005076 001406
1690 005100 016637 000002 001310
1691 005106 042766 000020 000002
1692 005114 000006
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706 005116
1707 005116 013766 001310 000002
1708 005124 042737 000020 001310
1709
1710 005132 000006
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723 005134 012703 170200
1724 005140 005023
1725 005142 022703 170376
1726 005146 103374
1727 005150 000207
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738 005152
1739 005152 005227
1740 005154 177777
1741 005156 001401
```

```

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

.SBTTL RESTORE T-BIT TO ITS PREVIOUS CONDITION
*****
*
* THIS SUBROUTINE CAN BE REACHED BY THE TRAP CALL 'TBITR', IT IS
* USED TO RESTORE THE T-BIT AFTER A PARTICULAR TEST THAT CANNOT
* BE RUN WITH THE T-BIT ON. IT USES THE PROCESSOR STATUS STORED
* IN 'OLDPSW' BY 'TBITO', REPLACES THE PS ON THE STACK WITH IT
* AND DOES AN 'RTT'.
*****
TBITRESTORE:
  MOV      OLDPSW,2(KSP)   ;PUT OLD PSW ON STACK
  BIC      #TBIT,OLDPSW    ;CLEAR T-BIT IN 'OLDPSW' SO THAT
                           ;IT WON'T BE TURNED ON BY ACCIDENT
  RTT              ;RETURN TO PROGRAM AND INHIBIT
                  ;T BIT TRAP AFTER THIS INSTRUCTION.

.SBTTL SUBROUTINE TO CLEAR ALL OF THE MAP REGISTERS
*****
*
* THIS SUBROUTINE CLEARS ALL OF THE MAP REGISTERS BY LOADING
* THE ADDRESS OF MAPLOO INTO R3 AND THEN CLEARING THE
* REGISTER POINTED TO BY R3 UNTIL R3 POINTS ABOVE MAPH37.
*****
CLRMAP: MOV      #MAPLO,R3    ;PUT FIRST MAP ADDR IN R3
1$:      CLR      (R3)+       ;CLEAR MAP REGS
         CMP      #MAPH37,R3  ;SEE IF LAST ADDR IS IN R3
         BHS      1$          ;BRANCH IF R3 LE THAN LAST ADDR
         RTS      PC          ;RETURN TO MAIN PROGRAM

.SBTTL SUBROUTINE TO LOG AND REPORT TIMEOUTS OF MAP REGISTERS
*****
*
* THIS SUBROUTINE IS USED TO LOG AND REPORT THE FACT THAT A
* REFERENCE TO A MAPPING REGISTER TIMED OUT ON THE UNIBUS. IT
* KEEPS A 'LOGICAL AND' AND A 'LOGICAL OR' OF EACH ADDRESS THAT
* TIMES OUT.
*****
TIMEOUT:
  INC      (PC)+             ;STARTING ADDRESS OF SUBROUTINE
  INCR     ONE TIME GATE
  TOFLAG: .WORD      -1       ;ONE TIME ENTANCE FLAG
         BEQ      10$         ;BRANCH IF FLAG IS NOW ZERO
```



```

1742 005160 000000          HALT          ; I HAVE ENTERED THIS ROUTINE BEFORE
1743                                     ; I FINISHED REPORTING THE FIRST ERROR
1744                                     ; THE SECOND ENTRY ADDRESS IS ON THE
1745                                     ; STACK AND THE FIRST ERROR CONDITION
1746                                     ; IS PROBABLY STILL LOCKED UP .
1747 005162 012637 001304    10$: MOV      (KSP)+,OLDPC      ;SAVE RETURN ADDRESS
1748 005166 012637 001306      MOV      (KSP)+,OLDPS      ;SAVE OLD PSW
1749 005172 013737 177766 001266      MOV      CPUERR,PCPUER ;SAVE CPU ERROR REGISTER
1750 005200 013737 001266 177766      MOV      PCPUER,CPUERR ;CLEAR CPU ERROR REGISTER
1751 005206 023737 001266 001264      CMP      PCPUER,CPUEXP ;SEE IF EXPECTED CONDITION CAME UP.
1752 005214 001402          BEQ        1$                ;BRANCH IF IT WAS A TIMEOUT
1753 005216 104001          ERROR      1                ;NOT RIGHT CONDITION
1754 005220 000414          BR         3$                ;BRANCH TO EXIT
1755 005222 050037 001226    1$: BIS      R0,ADDROR      ;PERFORM LOGIAL OR OF FAILING ADDRESS
1756 005226 005100          COM        R0                ;GET R0 READY FOR AND
1757 005230 040037 001224      BIC      R0,ADRAND        ;PERFORM LOGICAL AND
1758 005234 005100          COM        R0                ;PUT R0 BACK AS IT WAS
1759 005236 005737 001254      TST      ERRCNT          ;IS THIS THE FIRST ERROR
1760 005242 001002          BNE        2$                ;BRANCH IF NOT FIRST ERROR
1761 005244 104201          ERROR      201              ;NO REGISTER RESPONSE.
1762 005246 000401          BR         3$                ;BRANCH TO EXIT
1763 005250 104301          2$: ERROR      301          ;CONTINUE NO RESPONSE TABLE
1764 005252 012737 177777 005154    3$: MOV      #-1,TOFLAG ;RESET ONE TIME GATE
1765 005260 013746 001306      MOV      OLDPS,-(KSP)    ;RESTORE OLD PSW
1766 005264 013746 001304      MOV      OLDPC,-(KSP)    ;PUSH RETURN ADDRESS BACK ON THE STACK
1767 005270 000006          RTT                    ;RETURN TO THE TEST
1768
1769      .SBTTL  SUBROUTINE TO REPORT MAP REGISTERS THAT WILL NOT HOLD ZERO
1770      ;*****
1771      ;
1772      ; THIS SUBROUTINE IS CALLED EVERY TIME A MAP REGISTER IS CLEARED
1773      ; AND FOUND TO NOT BE ZERO.  IT KEEPS A LOGICAL 'AND' AND 'OR' OF
1774      ; THE FAILING REGISTER'S ADDRESS AND DATA AND REPORTS ALL ERRORS.
1775      ; AT THE END OF THE TEST A SUMMARY ERROR MESSAGE IS GIVEN WHICH
1776      ; WILL REPORT THE LOGICAL 'AND' AND 'OR' OF THE ADDRESSES AND DATA.
1777      ;
1778      ;*****
1779      WRITEERROR:
1780 005272 012637 001304      MOV      (KSP)+,OLDPC      ;SAVE RETURN ADDRESS IN CASE OF ERROR LOOP
1781 005276 050037 001226      BIS      R0,ADDROR      ;LOGICAL 'OR' OF BAD ADDRESS
1782 005302 005100          COM        R0                ;GET R0 READY FOR AND
1783 005304 040037 001224      BIC      R0,ADRAND        ;PERFORM LOGICAL AND
1784 005310 005100          COM        R0                ;PUT R0 BACK AS IT WAS
1785 005312 053737 001170 001232      BIS      $TMP0,DATAOR ;LOGICAL 'OR' OF BAD DATA
1786 005320 005137 001170          COM      $TMP0        ;GET $TMP0 READY FOR AND
1787 005324 043737 001170 001230      BIC      $TMP0,DATAND ;PERFORM LOGICAL AND
1788 005332 005137 001170          COM      $TMP0        ;PUT $TMP0 BACK AS IT WAS
1789 005336 005737 001254      TST      ERRCNT          ;SEE IF THIS IS FIRST ERROR
1790 005342 001002          BNE        1$                ;BRANCH IF NOT FIRST ERROR
1791 005344 104202          ERROR      202              ;ERROR TYPE OUT ITEM 4
1792 005346 000401          BR         2$                ;SKIP NEXT STATEMENT
1793 005350 104302          1$: ERROR      302              ;ERROR TYPE OUT ITEM 5
1794 005352 000177 173726    2$: JMP      @OLDPC        ;RETURN TO THE TEST
1795
1796      .SBTTL  SUBROUTINE TO REPORT DUAL ADDRESSING WHEN LOADING A MAP REGISTER
1797      ;*****

```

1798
1799
1800
1801
1802
1803
1804
1805
1806 005356
1807 005356 012637 001304
1808 005362 050037 001226
1809 005366 005100
1810 005370 040037 001224
1811 005374 005100
1812 005376 050137 001232
1813 005402 005101
1814 005404 040137 001230
1815 005410 005101
1816 005412 005737 001254
1817 005416 001002
1818 005420 104203
1819 005422 000401
1820 005424 104303
1821 005426 000177 173652
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833 005432 012637 001304
1834 005436 050037 001226
1835 005442 005100
1836 005444 040037 001224
1837 005450 005100
1838 005452 050337 001232
1839 005456 005103
1840 005460 040337 001230
1841 005464 005103
1842 005466 050437 001236
1843 005472 005104
1844 005474 040437 001234
1845 005500 005104
1846 005502 005737 001254
1847 005506 001002
1848 005510 104204
1849 005512 000401
1850 005514 104304
1851 005516 000177 173562
1852
1853

```

:
:
:      THIS SUBROUTINE WILL LOG AND REPORT ALL DUAL ADDRESSING ERRORS
:      FOUND IN THE MAPPING REGISTERS.  A 'LOGICAL OR' AND A
:      'LOGICAL AND' OF THE WRITTEN ADDRESSES WILL BE MAINTAINED IN
:      'ADDROR' AND 'ADRAND'.  THE LOG ON THE ADDITIONAL OR FAILING,
:      ADDRESSES WILL BE MAINTAINED IN 'DATAOR' AND 'DATAND'.
:
:*****
DUALADR:
      MOV      (KSP)+,OLDPC      ;STARTING LOCATION OF SUBROUTINE
      BIS      R0,ADDROR        ;SAVE RETURN ADDRESS IN CASE OF LOOP
      COM      R0                ;LOGICAL OR OF WRITTEN ADDRESS
      BIC      R0,ADRAND        ;GET R0 READY FOR AND
      COM      R0                ;PERFORM LOGICAL AND
      BIS      R1,DATAOR        ;PUT R0 BACK AS IT WAS
      COM      R1                ;LOGICAL OR OF DUALED ADDRESS
      BIC      R1,DATAND        ;GET R1 READY FOR AND
      COM      R1                ;PERFORM LOGICAL AND
      TST      ERRCNT           ;PUT R1 BACK AS IT WAS
      BNE      1$              ;SEE IF THIS IS FIRST ERROR
      ERROR    203              ;BRANCH IF NOT FIRST ERROR
      BR       2$              ;BRANCH TO EXIT
1$:      ERROR    303
2$:      JMP      @OLDPC        ;RETURN TO TEST
:
: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 SUMMARY IS GIVEN.
:
:*****
COUNT:  MOV      (KSP)+,OLDPC      ;SAVE RETURN ADDRESS IN CASE OF ERROR LOOP
      BIS      R0,ADDROR        ;LOGICAL OR OF BAD ADDRESSES
      COM      R0                ;GET R0 READY FOR AND
      BIC      R0,ADRAND        ;PERFORM LOGICAL AND
      COM      R0                ;PUT R0 BACK AS IT WAS
      BIS      R3,DATAOR        ;LOGICAL OR OF BAD DATA
      COM      R3                ;GET R3 READY FOR AND
      BIC      R3,DATAND        ;PERFORM LOGICAL AND
      COM      R3                ;PUT R3 BACK AS IT WAS
      BIS      R4,PATTOR        ;LOGICAL OR OF PATTERN LOADED
      COM      R4                ;GET R4 READY FOR AND
      BIC      R4,PATAND        ;PERFORM LOGICAL AND
      COM      R4                ;PUT R4 BACK AS IT WAS
      TST      ERRCNT           ;IS THIS FIRST ERROR
      BNE      1$              ;BRANCH IF NOT FIRST ERROR
      ERROR    204              ;REPORT FIRST COUNT ERROR
      BR       2$              ;SKIP NEXT STATEMENT
1$:      ERROR    304              ;REPORT MORE DATA
2$:      JMP      @OLDPC        ;RETURN TO THE TEST

```


1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864 005522 012637 001304
1865 005526 050137 001236
1866 005532 005101
1867 005534 040137 001234
1868 005540 005101
1869 005542 050037 001232
1870 005546 005100
1871 005550 040037 001230
1872 005554 005100
1873 005556 005737 001254
1874 005562 001002
1875 005564 104205
1876 005566 000401
1877 005570 104305
1878 005572 000177 173506
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892 005576
1893 005576 012637 001304
1894 005602 050237 001236
1895 005606 005102
1896 005610 040237 001234
1897 005614 005102
1898 005616 050337 001232
1899 005622 005103
1900 005624 040337 001230
1901 005630 005103
1902 005632 105737 001103
1903 005636 001002
1904 005640 104207
1905 005642 000401
1906 005644 104307
1907 005646 000177 173432
1908
1909

.SBTTL SUBROUTINE TO REPORT COUNT PATTERN ERRORS ON UNIBUS DATA PATH

*
* THIS SUBROUTINE IS CALLED WHENEVER AN ERROR IS DISCOVERED IN THE
* COUNT PATTERN THAT IS RUN OVER THE UNIBUS INTO MAIN MEMORY. IT
* KEEPS THE LOGICAL 'AND' AND 'OR' OF THE PATTERN LOADED AND THE
* DATA RECEIVED, AND REPORTS ALL ERRORS. AT THE END OF THE TEST
* IT GIVES A SUMMARY MESSAGE OF ALL THE ERRORS.

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

.SBTTL SUBROUTINE TO REPORT COUNT PATTERN ERRORS IN CACHE REGISTERS

*
* THIS SUBROUTINE IS USED TO LOG AND REPORT THE COUNT PATTERN
* ERRORS OCCURRING WHEN TESTING THE CACHE REGISTERS.
* THE 'LOGICAL OR' AND 'LOGICAL AND' OF VARIOUS DATA WILL BE
* MAINTAINED AS FOLLOWS:
* DATA FETCHED FROM REGISTERS IN 'DATAOR' AND 'DATAND'
* PATTERN LOADED INTO THE REGISTERS IN 'PATTOR' AND 'PATAND'.

CACOUNT:MOV (KSP)+,OLDPC ;STARTING ADDRESS OF THIS SUBROUTINE
BIS R2,PATTOR ;SAVE RETURN ADDRESS IN CASE OF LOOP
COM R2 ;LOGICAL OR OF PATTERN LOADED
BIC R2,PATAND ;GET R2 READY FOR AND
COM R2 ;PERFORM LOGICAL AND
BIS R3,DATAOR ;PUT R2 BACK AS IT WAS
COM R3 ;LOGICAL OR OF DATA FETCHED
BIC R3,DATAND ;GET R3 READY FOR AND
COM R3 ;PERFORM LOGICAL AND
TSTB \$ERFLG ;PUT R3 BACK AS IT WAS
BNE 1\$;SEE IF THIS IS THE FIRST ERROR
ERROR 207 ;BRANCH IF NOT FIRST ERROR
BR 2\$;BRANCH TO EXIT
1\$: ERROR 307
2\$: JMP @OLDPC ;RETURN TO TEST

PDP-11/70-74MP UNIBUS MAP DIAGNOSTIC
CEKBFC.P11 03-MAY-79 13:19

MACY11 30A(1052) 03-MAY-79^{J 4} 13:21 PAGE 37
***** TRAP HANDLING ROUTINES *****

SEQ 0048

1910
1911
1912
1913
1914
1915
1916

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

.SBTTL CPU TRAP HANDLER ROUTINE

.*

.*

.*

THIS SUBROUTINE WILL HANDLE ALL CPU TRAPS AND ABORTS, THRU
'ERRVEC' (000004). IF THIS SUBROUTINE IS ENTERED BY A SECOND

1917
1918
1919
1920

:* TRAP BEFORE THE FIRST HAS BEEN PROCESSED A HALT IS EXECUTED.
:* IF THE WORD 'CPUEXP' IS ZERO, NO TRAP WAS EXPECTED AND AN
:* UNEXPECTED ERROR MESSAGE IS GIVEN. IF THE WORD 'CPUEXP' IS
:* NOT ZERO THEN THE CPU ERROR REGISTER 'CPUERR' IS COMPARED WITH

1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931

```

;*      'CPUEXP' TO SEE IF THE PROPER CONDITION OCCURRED. 'PCPUER' CAN
;*      BE USED AS A FLAG TO INDICATE THAT A TRAP HAS OCCURRED SINCE IT
;*      IS LOADED WITH THE ERROR REGISTER IF A TRAP VECTORS HERE

```

1926	005652	005227
1927	005654	177777
1928	005656	001401
1929	005660	000000

```
CPUER: INC (PC)+ ;MAKE FLAG ZERO IF FIRST TIME
CPFLAG: .WORD -1 ;NEGATIVE ONE FOR A FLAG
        BEQ 10$ ;BRANCH IF FIRST TIME IN
        HALT ;I HAVE ENTERED THIS ROUTINE BEFORE
           ;I FINISHED REPORTING THE FIRST ERROR
           ;THE SECOND ENTRY ADDRESS IS ON THE
```



```
1932                                     ;STACK AND THE FIRST ERROR CONDITION
1933                                     ;IS PROBABLY STILL LOCKED UP
1934 005662 012637 001304 10$: MOV (KSP)+,OLDPC ;SAVE RETURN ADDRESS IN CASE OF LOOP
1935 005666 012637 001306      MOV (KSP)+,OLDPS ;SAVE OLD PSW IN CASE OF LOOP
1936 005672 012706 001100      MOV #KERSTK,KSP ;MAKE SURE THAT THE KERNEL STACK POINTER
1937                                     ;IS AT 1100.
1938 005676 013737 177766 001266      MOV CPUERR,PCPUER ;SAVE CPU ERROR REGISTER
1939 005704 013737 001304 001302      MOV OLDPC,BADPC ;SAVE PC+2 AT TIME OF ABORT
1940 005712 005737 001264      TST CPUEXP ;SEE IF ANY CONDITION WAS EXPECTED
1941 005716 001406      BEQ 2$ ;BRANCH IF NO TRAP WAS EXPECTED
1942 005720 023737 001266 001264      CMP PCPUER,CPUEXP ;SEE IF EXPECTED ERROR OCCURED
1943 005726 001403      BEQ 1$ ;BRANCH IF ERROR CODES MATCH
1944 005730 104001      ERROR 1 ;WRONG CPU TRAP CONDITION
1945 005732 000401      BR 1$ ;SKIP NEXT INSTRUCTION
1946 005734 104002 2$: ERROR 2 ;NO CPU TRAP EXPECTED
1947 005736 013737 001266 177766 1$: MOV PCPUER,CPUERR ;CLEAR CPU ERROR REGISTER
1948 005744 012737 177777 005654      MOV #-1,CPFLAG ;MAKE FLAG NEGATIVE ONE FOR NEXT TIME
1949 005752 013746 001306      MOV OLDPS,-(KSP) ;PUSH OLD PSW BACK ON STACK
1950 005756 013746 001304      MOV OLDPC,-(KSP) ;PUSH RETURN ADDRESS BACK ON STACK
1951 005762 000006      RTT ;RETURN FROM INTERRUPT OR ABORT
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
```

.SBTTL CACHE TRAPS AND ABORTS HANDLER ROUTINE

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

```
1968 005764 005227 MEMER: INC (PC)+ ;MAKE FLAG ZERO IF FIRST TIME
1969 005766 177777 PAFLAG: .WORD -1 ;NEGATIVE ONE FOR A FLAG
1970 005770 001401      BEQ 10$ ;BRANCH IF FIRST TIME IN
1971 005772 000000      HALT ;I HAVE ENTERED THIS ROUTINE BEFORE
1972                                     ;I FINISHED REPORTING THE FIRST ERROR
1973                                     ;THE SECOND ENTRY ADDRESS IS ON THE
1974                                     ;STACK AND THE FIRST ERROR CONDITION
1975                                     ;IS PROBABLY STILL LOCKED UP
1976 005774 012737 006206 000114 10$: MOV #5$,CACHVEC ;SET CACHE VECTOR TO RTI UNTIL THE
1977                                     ;THE CACHE HAS BEEN FIXED.
1978 006002 012637 001304      MOV (KSP)+,OLDPC ;SAVE RETURN ADDRESS IN CASE OF LOOP
1979 006006 012637 001306      MOV (KSP)+,OLDPS ;SAVE OLD PSW IN CASE OF LOOP
1980 006012 013737 177740 001270      MOV @#LOADRS,PLOADR ;SAVE LOWER CACHE ADDR REG
1981 006020 013737 177742 001272      MOV @#HIADRS,PHIADR ;SAVE HIGH BITS OF FAILING ADDR
1982 006026 013737 177744 001274      MOV @#MEMERR,PPARER ;SAVE MEMORY ERROR REGISTER
1983 006034 013737 177746 001276      MOV @#CONTRL,PCONTR ;SAVE CONTROL REGISTER FOR TYPE OUT
1984 006042 013737 177750 001300      MOV @#MAINT,PMaint ;SAVE MAINTENANCE REGISTER
1985 006050 013737 001304 001302      MOV OLDPC,BADPC ;SAVE PC+2 AT TIME OF ABORT
1986 006056 032737 000014 001274      BIT #14,PPARER ;WAS THIS A MAIN MEMORY REFERENCE
1987 006064 001005      BNE 1$ ;BRANCH IF MAIN MEMORY PARITY ERROR
```

```
1988 006066 012737 005764 000114      MOV    #MEMER,CACHVEC ;LOAD ADDRESS OF THIS ROUTINE IN VECTOR
1989 006074 104003                      ERROR    3          ;UNEXPECTED PARITY ABORT
1990 006076 000415                      BR        2$          ;BRANCH TO SUBROUTINE EXIT
1991 006100 010046                      1$:    MOV    R0,-(KSP)      ;SAVE R0 ON STACK
1992 006102 013700 001270                MOV    PLOADR,R0      ;PUT LOW 16 BITS OF BAD ADDR IN R0
1993 006106 042700 176000                BIC     #176000,R0      ;CLEAR UPPER 6 BITS OF ADDRESS
1994 006112 074027 000002                XOR     R0,#BIT1      ;REFERENCE OTHER WORD OF PAIR
1995 006116 005710                      TST     (R0)          ;READ AT SAME INDEX AS BAD WORD
1996 006120 012600                      MOV    (KSP)+,R0      ;RESTORE R0 FROM STACK
1997 006122 012737 005764 000114        MOV    #MEMER,CACHVEC ;LOAD ADDRESS OF THIS ROUTINE IN VECTOR
1998 006130 104004                      ERROR    4          ;MAIN MEMORY PARITY ERROR
1999 006132 005737 001322                2$:    TST     RETRY      ;ARE YOU RETRYING THIS TEST?
2000 006136 001006                      BNE     3$          ;BRANCH IF THIS IS SECOND TRY
2001 006140 005237 001322                INC     RETRY      ;SET RETRY FLAG
2002 006144 013737 001106 001304        MOV    $LPADR,OLDPC  ;RETURN TO START OF THIS TEST
2003 006152 000403                      BR        4$          ;BRANCH TO EXIT
2004 006154 013737 001324 001304        3$:    MOV    NXTTST,OLDPC ;RETURN TO START OF NEXT TEST
2005                                ;SINCE THIS IS THE SECOND ABORT
2006 006162 013737 001274 177744        4$:    MOV    PPARER,MEMERR ;CLEAR MEMORY ERROR REGISTER
2007 006170 012737 177777 005766        MOV    #-1,PAFLAG  ;RESTORE A NEGATIVE ONE FOR NEXT TIME
2008 006176 013746 001306                MOV    OLDPS,-(KSP)  ;PUSH OLD PSW BACK ON STACK
2009 006202 013746 001304                MOV    OLDPC,-(KSP)  ;PUSH RETURN ADDRESS BACK ON STACK
2010 006206 000006                5$:    RTT          ;RETURN FROM INTERRUPT.
```

.SBTTL MEMORY MANAGEMENT TRAPS AND ABORTS HANDLER ROUTINE

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

```
2023 006210 005227      MMTRAP: INC    (PC)+      ;MAKE FLAG ZERO IF FIRST TIME
2024 006212 177777      MMFLAG: .WORD  -1        ;FLAG SHOULD BE NEG ONE
2025 006214 001401                      BEQ     10$      ;BRANCH IF FIRST TIME INTO ROUTINE
2026 006216 000000                      HALT          ;I HAVE ENTERED THIS ROUTINE BEFORE
2027                                ;I FINISHED REPORTING THE FIRST ERROR
2028                                ;THE SECOND ENTRY ADDRESS IS ON THE
2029                                ;STACK AND THE FIRST ERROR CONDITION
2030                                ;IS PROBABLY STILL LOCKED UP
2031 006220 011637 001302      10$:    MOV    (KSP),BADPC    ;SAVE PC AT TIME OF ABORT OR TRAP
2032 006224 012637 001304                MOV    (KSP)+,OLDPC  ;SAVE RETURN ADDRESS IN CASE OF LOOP
2033 006230 012637 001306                MOV    (KSP)+,OLDPS  ;SAVE OLD PSW IN CASE OF LOOP
2034 006234 013737 177572 001312        MOV    MMR0,PMMR0    ;SAVE STATUS REGISTER
2035 006242 013737 177574 001314        MOV    MMR1,PMMR1    ;SAVE AUTO INC/DEC REGISTER
2036 006250 013737 177576 001316        MOV    MMR2,PMMR2    ;SAVE VIRTUAL ADDRESS REGISTER
2037 006256 104005                      ERROR    5          ;UNEXPECTED M.M. ABORT OR TRAP
2038 006260 042737 177776 177572        1$:    BIC     #177776,@MMR0 ;CLEAR ALL BITS EXCEPT 0
2039 006266 012737 177777 006212        MOV    #-1,MMFLAG  ;RESTORE A NEGATIVE ONE TO FLAG
2040 006274 013746 001306                MOV    OLDPS,-(KSP)  ;PUSH OLD PSW ONTO STACK
2041 006300 013746 001304                MOV    OLDPC,-(KSP)  ;PUSH RETURN ADDRESS ON STACK
2042 006304 000006                RTT          ;RETURN TO MAIN PROGRAM
2043
```



```
2044
2045
2046
2047
2048
2049      010000
2050
2051 010000
2052 010000 012737 000340 177776
2053 010006 012706 001100
2054 010012 005026
2055 010014 022706 001136
2056 010020 001374
2057 010022 012706 001100
2058 010026 012737 002214 000020
2059 010034 012737 000340 000022
2060 010042 012737 002534 000030
2061 010050 012737 000340 000032
2062 010056 012737 004506 000034
2063 010064 012737 000340 000036
2064 010072 012737 004550 000024
2065 010100 012737 000340 000026
2066 010106 013737 024414 024406
2067 010114 005037 001204
2068 010120 005037 001206
2069 010124 112737 000001 001115
2070 010132 012737 024634 000014
2071 010140 012737 000340 000016
2072 010146 012737 000002 024634
2073 010154 012737 010202 000010
2074 010162 005046
2075 010164 012746 010172
2076 010170 000006
2077 010172 012737 000006 024634 64$:
2078 010200 000402
2079 010202 062706 000010 65$:
2080 010206 012737 000012 000010 66$:
2081 010214 005037 024642
2082 010220 012737 010220 001106
2083 010226 012737 010226 001110
2084 010234 005227 177777
2085 010240 001037
2086 010242 022737 024566 000042
2087 010250 001433
2088 010252 104400 010260
2089 010256 000430
2090
2091 010340
2092
2093
2094
2095
2096
2097
2098
2099

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

START:
MOV      #340,@#PS      ;;LOCK OUT ALL INTERRUPTS
MOV      #SCMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
CLR      (R6)+           ;;CLEAR MEMORY LOCATION
CMP      #STKS,R6        ;;DONE?
BNE      .-6            ;;LOOP BACK IF NO
MOV      #STACK,SP       ;;SETUP THE STACK POINTER
MOV      #SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
MOV      #340,@#IOTVEC+2 ;;LEVEL 7
MOV      #ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
MOV      #340,@#EMTVEC+2 ;;LEVEL 7
MOV      #STRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
MOV      #340,@#TRAPVEC+2;LEVEL 7
MOV      #SPWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
MOV      #340,@#PWRVEC+2 ;;LEVEL 7
MOV      $ENDCT,$EOPCT   ;;SETUP END-OF-PROGRAM COUNTER
CLR      $TIMES          ;;INITIALIZE NUMBER OF ITERATIONS
CLR      $ESCAPE         ;;CLEAR THE ESCAPE ON ERROR ADDRESS
MOVB     #1,$ERMAX       ;;ALLOW ONE ERROR PER TEST
MOV      #SRTN,@#TBITVEC ;;SET 'T' BIT VECTOR TO SRTN
MOV      #340,@#TBITVEC+2;LEVEL 7
MOV      #RTI,$SRTN      ;;SET $SRTN TO A RTI
MOV      #65$,@#RESVEC   ;;TRY TO DO A RTT
CLR      -(SP)           ;;DUMMY PS
MOV      #64$,-(SP)      ;;AND PC
RTT              ;;TRY THE RTT
MOV      #RTT,$SRTN      ;;RTT IS LEGAL--SET $SRTN TO A RTT
BR      66$
ADD      #10,SP          ;;RTT ILLEGAL--CLEAN OFF THE STACK
MOV      #RESVEC+2,@#RESVEC;;RESTORE TRAP CATCHER
CLR      $TBIT          ;;CLEAR 'T' BIT SWITCH
MOV      #.,$LPADR       ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
MOV      #.,$LPERR       ;;SETUP THE ERROR LOOP ADDRESS
INC      #-1            ;;FIRST TIME?
BNE      67$            ;;BRANCH IF NO
CMP      #SENDAD,@#42    ;;ACT-11?
BEQ      67$            ;;BRANCH IF YES
TYPE     ,68$           ;;TYPE ASCIZ STRING
BR      67$            ;;GET OVER THE ASCIZ
;;68$: .ASCIZ <CRLF>?CEKBF.C PDP-11/70-74MP UNIBUS MAP DIAGNOSTIC?<CRLF>
67$:
;;
;;*** TEST FOR VARIOUS KB11 PROCESSORS ***
;;
;;*THIS ROUTINE POLES THE RESULTS OF ATTEMPTS TO SET TO ONE
;;*CERTAIN CRITICAL BITS THAT ARE KNOWN TO BE OPERATIVE ON A KB11CM,
;;*OR KB11EM PROCESSOR. IF TWO OUT OF FOUR OF THE TESTS ARE
;;*POSITIVE THEN THE KB11CM OR KB11EM FLAG IS SET,IF LESS THAN TWO OF THE
;;*TESTS ARE POSITIVE THEN THE KB11E FLAG OR NO FLAG IS SET. THE DETERMINATION
```

```

2100      ;:OF WHICH PAIR IS VALID IS BASED ON THE RESULTS OF EXECUTING AN MFPT OPCODE
2101      ;:(OPCODE 7). IF THIS INSTRUCTION TRAPS THIS IS AN KB11CM OR
2102      ;:A PLAIN 1170 (KB11-B OR KB11-C). IF THE INSTRUCTION DOES NOT TRAP THEN
2103      ;:THIS IS A KB11-E OR KB11-EM.
2104
2105 010340 105037 001332      KBTST: CLRB      @#KB11CM      ;RESET THE MP FLAG
2106 010344 005037 001330      CLR      @#KB11E      ;CLEAR KB11E AND KB11EM FLAGS
2107 010350 012737 010606 000010      MOV      #MFPTTR,@#RESVEC ;SET UP TRAP ADDRESS FOR MFPT AT RESERV VECTOR
2108 010356 000007      MFPT      ;EXECUTE MFPT. WILL TRAP ON 1170 (KB11B/C) OR
2109      ;KB11CM (11/74)
2110 010360 012737 000001 001330      MOV      #1,@#KB11E      ;HERE IF KB11E OR KB11EM. SET FLAG
2111 010366 005037 177750      T1:      CLR      @#MAINT      ;CLEAR THE MAINTENANCE REGISTER
2112 010372 005005      CLR      R5      ;RESET THE TEST COUNTER
2113 010374 012700 177746      MOV      #CONTRL,R0      ;GET THE ADDRESS OF...
2114 010400 012701 177750      MOV      #MAINT,R1      ;CCR,MAINT,AND MAPH00...
2115 010404 012702 170202      MOV      #MAPH00,R2      ;AND PLACE IN R0-R2
2116 010410 052710 040000      BIS      #BIT14,(R0)      ;TRY TO SET IVSS BIT
2117 010414 032710 040000      BIT      #BIT14,(R0)      ;DID IT SET?
2118 010420 001403      BEQ      T2      ;NO,GO TO NEXT TEST
2119 010422 042710 040000      BIC      #BIT14,(R0)      ;CLEAR IT.
2120 010426 005205      INC      R5      ;TEST IS POSITIVE
2121 010430 052711 000001      T2:      BIS      #BIT0,(R1)      ;SET EDMA IN MAINT REGISTER
2122 010434 032711 000001      BIT      #BIT0,(R1)
2123 010440 001410      BEQ      T3
2124 010442 052710 004000      BIS      #BIT11,(R0)      ;TRY TO SET DMA IN CCR
2125 010446 032710 004000      BIT      #BIT11,(R0)
2126 010452 001403      BEQ      T3
2127 010454 042710 004000      BIC      #BIT11,(R0)
2128 010460 005205      INC      R5
2129 010462 042711 000001      T3:      BIC      #BIT0,(R1)      ;MAKE SURE EDMA IS CLEAR
2130 010466 052737 100000 172300      BIS      #BIT15,KIPDR0      ;TRY TO SET BYP ON A PDR
2131 010474 032737 100000 172300      BIT      #BIT15,KIPDR0
2132 010502 001404      BEQ      T4
2133 010504 042737 100000 172300      BIC      #BIT15,KIPDR0
2134 010512 005205      INC      R5
2135 010514 052712 100000      T4:      BIS      #BIT15,(R2)      ;TRY TO SET BYP ON UNIBUS MAP
2136 010520 032712 100000      BIT      #BIT15,(R2)
2137 010524 001403      BEQ      T.END
2138 010526 042712 100000      BIC      #BIT15,(R2)
2139 010532 005205      INC      R5
2140 010534 022705 000002      T.END:  CMP      #2,R5      ;IS THE RESULT OF THE TEST >=2
2141 010540 101021      BHI      2$      ;NO,THIS IT A KB11E OR KB11-B/C (11/70)
2142 010542 005000      CLR      R0
2143 010544 005037 177746      CLR      @#CONTRL
2144 010550 013701 177746      3$:      MOV      @#CONTRL,R1
2145 010554 001402      BEQ      4$
2146 010556 005200      INC      R0
2147 010560 001373      BNE      3$
2148 010562      4$:
2149 010562 005737 001330      TST      @#KB11E      ;IS IS A KB11-E OR KB11-EM?
2150 010566 001404      BEQ      1$      ;BR IF NEITHER. MUST BE KB11CM
2151 010570 012737 000400 001330      MOV      #BIT8,@#KB11E      ;SET UPPER BYTE (KB11-EM)
2152 010576 000402      BR      2$      ;DONE
2153 010600 105237 001332      1$:      INCB      @#KB11CM      ;YES, FLAG THIS AS A MODIFIED PROCESSOR
2154 010604 000403      2$:      BR      ENDKB      ;DONE DETERMINING WHICH CPU
2155

```



```

2156 010606      012716 010366      MFPTTR:      MOV      #T1,(SP)      ;HERE IF MFPT TRAPPED. SEE IF 1170 OR KB11CM
2157 010606      000002      RTI      ;SET UP RETURN ADDRESS FOR RTI
2158 010612      000002      ENDKB:      ;RETURN
2159 010614      005227 177777      INC      #-1      ;FIRST TIME?
2160 010614      001026      BNE      100$      ;BR IF NO
2161 010620      104400 024650      TYPE      ,MSG1      ;<15><12>CPU UNDER TEST FOUND TO BE A
2162 010622      005737 001330      TST      @#KB11E      ;IS THIS A KB11-E OR KB11-EM?
2163 010626      001011      BNE      101$      ;BR IF EITHER ONE
2164 010632      105737 001332      TSTB     @#KB11CM      ;IS IT A 11/74      (KB11CM)
2165 010634      001003      BNE      1$      ;BR IF IT IS
2166 010640      104400 024720      TYPE      ,MSG3      ;KB11-B/C<15><12>
2167 010642      000413      BR      100$      ;SKIP OTHER MESSAGE
2168 010646      104400 024732      1$:      TYPE      ,MSG4      ;11/74      (KB11CM)<15><12>
2169 010650      000410      BR      100$      ;SKIP CISP MESSAGE
2170 010654      105737 001330      101$:    TSTB     @#KB11E      ;IS IT A KB11-E?
2171 010656      001403      BEQ      102$      ;BR IF NOT. MUST BE KB11-EM
2172 010662      104400 024763      TYPE      ,MSG5      ;KB11-E<15><12>
2173 010664      000402      BR      100$      ;SKIP KB11-EM MESSAGE
2174 010670      104400 024707      102$:    TYPE      ,MSG2      ;KB11-EM<15><12>
2175 010672      000114      100$:
2176 010676      012737 005764 000114      LOOP:  MOV      #MEMER,CACHVEC ;LOAD PARITY ERROR SERVICE
2177 010676      012737 000340 000116      MOV      #340,CACHVEC+2 ;ROUTINE ADDRESS
2178 010676      012737 006210 000250      MOV      #MMTRAP,MMVEC ;SET PRIORITY SEVEN
2179 010704      012737 000340 000116      MOV      #MMTRAP,MMVEC ;LOAD MEMORY MANAGEMENT TRAP
2180 010712      012737 006210 000250      MOV      #MMTRAP,MMVEC ;SERVICE ROUTINE ADDRESS
2181 010720      012737 000340 000252      MOV      #340,MMVEC+2 ;SET PRIORITY SEVEN
2182 010726      012737 005652 000004      MOV      #CPUER,ERRVEC ;LOAD CPU TRAP SERVICE ROUTINE ADDR
2183 010726      012737 005652 000004      MOV      #CPUER,ERRVEC ;SET PRIORITY SEVEN
2184 010734      012737 000340 000006      MOV      #340,ERRVEC+2 ;LOAD ROM ADDRESS OF 'NOP' INTO MICRO
2185 010742      012727 000044 177770      MOV      #044,#177770 ;BREAK REGISTER. EVERY 'NOP' WILL
2186 010742      012727 000044 177770      MOV      #044,#177770 ;GENERATE A 'SYNC' PULSE ON PIN
2187 010742      012727 000044 177770      MOV      #044,#177770 ;#AE1 SLOT 10
2188 010742      012727 000044 177770      MOV      #044,#177770 ;NOT EXPECTING ANY CPU ERRORS
2189 010750      005037 001264      CLR      CPUEXP      ;START IN 16 BIT MAPPING
2190 010754      005037 177572      CLR      @#MMR0      ;DISABLE MAP AND 22-BIT MAPPING
2191 010760      005037 172516      CLR      @#MMR3      ;NEGATIVE ONE USED TO INITIALIZE FLAGS
2192 010764      012700 177777      MOV      #-1,R0      ;INITIALIZE FLAGS
2193 010770      010037 005654      MOV      R0,CPFLAG      ;INITIALIZE FLAGS
2194 010774      010037 005154      MOV      R0,TOFLAG      ;INITIALIZE FLAGS
2195 011000      010037 005766      MOV      R0,PAFLAG      ;INITIALIZE FLAGS
2196 011004      010037 006212      MOV      R0,MMFLAG      ;INITIALIZE FLAGS
2197 011010      010037 177744      MOV      R0,@#MEMERR      ;CLEAR PARITY ERROR REGISTER
2198 011014      010037 177766      MOV      R0,@#CPUERR      ;CLEAR CPU ERROR REGISTER
2199 011014      010037 177766      MOV      R0,@#CPUERR      ;CLEAR CPU ERROR REGISTER
2200 011014      010037 177766      MOV      R0,@#CPUERR      ;CLEAR CPU ERROR REGISTER
2201 011014      010037 177766      MOV      R0,@#CPUERR      ;CLEAR CPU ERROR REGISTER
2202 011014      010037 177766      MOV      R0,@#CPUERR      ;CLEAR CPU ERROR REGISTER
2203 011014      010037 177766      MOV      R0,@#CPUERR      ;CLEAR CPU ERROR REGISTER
2204 011014      010037 177766      MOV      R0,@#CPUERR      ;CLEAR CPU ERROR REGISTER
2205 011014      010037 177766      MOV      R0,@#CPUERR      ;CLEAR CPU ERROR REGISTER
2206 011014      010037 177766      MOV      R0,@#CPUERR      ;CLEAR CPU ERROR REGISTER
2207 011014      010037 177766      MOV      R0,@#CPUERR      ;CLEAR CPU ERROR REGISTER
2208 011014      010037 177766      MOV      R0,@#CPUERR      ;CLEAR CPU ERROR REGISTER
2209 011014      010037 177766      MOV      R0,@#CPUERR      ;CLEAR CPU ERROR REGISTER
2210 011014      010037 177766      MOV      R0,@#CPUERR      ;CLEAR CPU ERROR REGISTER
2211 011014      010037 177766      MOV      R0,@#CPUERR      ;CLEAR CPU ERROR REGISTER

```

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

;*****
 ;*TEST 1 MAP REGISTER RESPONSE TEST
 ;*
 ;* THIS TEST IS USED TO ENSURE THAT ALL THE UNIBUS MAP REGISTERS
 ;* CAN BE REFERENCED UNDER PROGRAM CONTROL, WITHOUT TIMING OUT.

```
2212      ;* THE ADDRESSES OF ANY MAP REGISTERS THAT TIME OUT WILL BE REPORTED
2213      ;* AND, AT THE END OF THE TEST, A SUMMARY OF THOSE REGISTERS WILL
2214      ;* BE GIVEN. THE ADDRESS DECODE CIRCUITRY IS ON 'MAPB' AND 'MAPC'.
2215      ;*
2216      ;*****
2217      TST1:
2218      SCOPE
2219      MOV      #TST2,NXTTST      ;SAVE STARTING ADDRESS OF NEXT TEST
2220      ;FOR ESCAPE ON PARITY ERRORS
2221      20$:    MOV      #TIMEOUT,ERRVEC ;LOAD ERRVEC WITH ROUTINE ADDR.
2222      MOV      #MAPLO,R0      ;PUT FIRST MAP REG ADDR IN R0
2223      MOV      #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
2224      1$:    NOP
2225      MOV      (R0),R2      ;THIS IS A SYNC POINT FOR SCOPING
2226      ADD      #2,R0      ;READ MAP REGISTERS TO R2
2227      CMP      #MAPH37,R0      ;POINT TO NEXT MAP REGISTER
2228      BHS      1$      ;COMPARE LAST MAP REG ADDR WITH R0
2229      MOV      #20$,$LPERR      ;CONTINUE READING IF NOT AT LAST REG.
2230      ;INITIALIZE LOOPING POINTER IN CASE
2231      TST      ERRCNT      ;OF INTERMITTENT FAULTS.
2232      BEQ      TST2      ;SEE IF THERE WERE ANY ERRORS
2233      ERROR    6      ;NO ERRORS GO TO NEXT TEST
2234      ;SUMMARY OF MAP REGISTERS THAT TIMED OUT
2235      ;*****
2236      ;*TEST 2      CACHE REGISTER RESPONSE TEST
2237      ;*****
2238      ;*
2239      ;* THIS TEST IS USED TO ENSURE THAT ALL THE CACHE REGISTERS
2240      ;* CAN BE REFERENCED UNDER PROGRAM CONTROL, WITHOUT TIMING OUT.
2241      ;* THE ADDRESSES OF ANY CACHE REGISTERS THAT TIME OUT WILL BE
2242      ;* REPORTED AND, AT THE END OF THE TEST, A SUMMARY OF THOSE REGISTERS
2243      ;* WILL BE GIVEN.
2244      ;* THESE REGISTERS ARE TESTED HERE BECAUSE THE ADDRESS DECODE AND DATA
2245      ;* PATH IS ON THE UNIBUS MAP BOARD (8141). THE ADDRESS DECODE CIRCUITRY
2246      ;* IS ON 'MAPB' AND GENERATES 'MAPB CACHE REG'.
2247      ;*
2248      ;*****
2249      TST2:
2250      SCOPE
2251      MOV      #TST3,NXTTST      ;SAVE STARTING ADDRESS OF NEXT TEST
2252      ;FOR ESCAPE ON PARITY ERRORS
2253      20$:    MOV      #LOADRS,R0      ;PUT ADDR OF FIRST CACHE REG IN R0
2254      MOV      #1$,$LPERR      ;SET LOOP ON ERROR POINTER HERE
2255      1$:    NOP
2256      MOV      (R0),R2      ;THIS IS A SYNC POINT FOR SCOPING
2257      ADD      #2,R0      ;TRY TO READ ALL CACHE REGISTERS
2258      CMP      #HITMIS,R0      ;POINT TO NEXT CACHE REGISTER
2259      BHS      1$      ;HAVE ALL REGISTERS BEEN REFERENCED?
2260      MOV      #20$,$LPERR      ;BRANCH IF MORE TO TEST
2261      ;INITIALIZE LOOPING ERROR IN CASE
2262      TST      ERRCNT      ;OF INTERMITTENT FAULTS
2263      BEQ      TST3      ;WERE THERE ANY ERRORS.
2264      ERROR    7      ;BRANCH IF NO ERRORS
2265      ;SUMMARY OF CACHE REGISTERS THAT TIMED OUT
2266      ;*THE NEXT FOUR (4) TESTS ARE USED TO CHECK THE GENERATION
2267      ;*OF: 'MAPB WRITE HI WORD L' AND 'MAPB WRITE LOWORD L'.
```



```

2268 ;*ALL REGISTERS ARE CLEARED AND THEN READ AND CHECKED FOR
2269 ;*ZEROS. THESE TESTS ALSO DETECT BITS IN THE DATA PATH
2270 ;*TO AND FROM THE MAP REGISTERS OR IN THE MAP REGISTERS STUCK
2271 ;*AT '1'. THE RECEIVERS ON PAGE MAPA AND THE B & C INPUTS
2272 ;*TO THE MULTIPLEXERS ON PAGE MAPJ AND THE DRIVERS ON PAGE
2273 ;*MAPJ MAKE UP THE DATA PATH UNDER TEST HERE.
2274
2275 ;*****
2276 ;*TEST 3 CLEAR AND READ LOW 20 REGISTERS LOWER 16 BITS
2277 ;*
2278 ;* THIS TEST WILL ENSURE THAT MAPL00 - MAPL17 WILL ALL HOLD ZERO.
2279 ;* IF ANY REGISTERS ARE NOT ZERO AFTER BEING CLEARED, THEIR ADDRESS AND
2280 ;* CONTENTS ARE REPORTED. IF ALL THE REGISTERS HAVE RANDOM DATA THEN
2281 ;* PROBABLY NO WRITE PULSE IS BEING GENERATED, BUT IF THE ERROR IS IN
2282 ;* ONLY ONE REGISTER OR IN ONE BIT IT MIGHT BE MORE DIFFICULT TO FIND.
2283 ;* IT IS POSSIBLE THAT THE COUNT PATTERN TEST FOR MAPL00 - MAPL17 WILL
2284 ;* GIVE MORE HELP.
2285 ;*
2286 ;*****
2287 011162 TST3:
2288 011162 000004 SCOPE
2289 011164 012737 011260 001324 MOV #TST4,NXTTST ;SAVE STARTING ADDRESS OF NEXT TEST
2290 ;*
2291 011172 012737 005652 000004 MOV #CPUER,ERRVEC ;FOR ESCAPE ON PARITY ERRORS
2292 011200 012700 170200 20$: MOV #MAPL0,R0 ;LOAD CPU TRAP SERVICE ROUTINE ADDR
2293 011204 012737 011212 001110 MOV #1$, $LPERR ;LOAD STARTING MAP REGISTER ADDR
2294 011212 000240 1$: NOP ;SET LOOP ON ERROR POINTER HERE
2295 011214 005010 CLR (R0) ;THIS IS A SYNC POINT FOR SCOPING
2296 011216 011037 001170 MOV (R0), $TMP0 ;CLEAR MAP REGISTER
2297 011222 001402 BEQ 2$ ;SEE IF MAP REGISTER IS ZERO
2298 011224 004737 005272 JSR PC,WRITEERROR ;BRANCH IF REGISTER IS ZERO
2299 011230 062700 000004 2$: ADD #4,R0 ;REPORT AND LOG ERROR
2300 011234 022700 170274 CMP #MAPL17,R0 ;POINT TO NEXT MAP REG LOWER 16 BITS
2301 011240 103364 BHS 1$ ;SEE IF THIS SECTION IS TESTED
2302 011242 012737 011200 001110 MOV #20$, $LPERR ;BRANCH IF NOT
2303 ;*
2304 011250 005737 001254 TST ERRCNT ;INITIALIZE LOOPING POINTER IN CASE
2305 011254 001401 BEQ TST4 ;OF INTERMITTENT FAULTS.
2306 011256 104010 ERROR 10 ;WERE THERE ANY ERRORS ON THIS TEST
2307 ;*
2308 ;*
2309 ;*
2310 ;*
2311 ;*
2312 ;*
2313 ;*
2314 ;*
2315 ;*
2316 ;*
2317 ;*
2318 ;*
2319 ;*
2320 ;*
2321 ;*
2322 011260 TST4:
2323 011260 000004 SCOPE

```

```

;*****
;*TEST 4 CLEAR AND READ LOW 20 REGISTERS UPPER 6 BITS
;*
;* THIS TEST WILL ENSURE THAT MAPH00 - MAPH17 WILL ALL HOLD ZERO.
;* IF ANY REGISTERS ARE NOT ZERO AFTER BEING CLEARED, THEIR ADDRESS AND
;* CONTENTS ARE REPORTED. IF ALL THE REGISTERS HAVE RANDOM DATA THEN
;* PROBABLY NO WRITE PULSE IS BEING GENERATED, BUT IF THE ERROR IS IN
;* ONLY ONE REGISTER OR IN ONE BIT IT MIGHT BE MORE DIFFICULT TO FIND.
;* IT IS POSSIBLE THAT THE COUNT PATTERN TEST FOR MAPH00 - MAPH17 WILL
;* GIVE MORE HELP.
;*
;*****
TST4:
SCOPE

```

```
2324 011262 012737 011350 001324      MOV      #TST5,NXTTST      ;SAVE STARTING ADDRESS OF NEXT TEST
2325                                     ;FOR ESCAPE ON PARITY ERRORS
2326 011270 012700 170202      20$: MOV      #MAPH0,R0      ;LOAD STARTING MAP REGISTER ADDR
2327 011274 012737 011302 001110      MOV      #1$,$LPERR      ;SET LOOP ON ERROR POINTER HERE
2328 011302 000240      1$: NOP                                     ;THIS IS A SYNC POINT FOR SCOPING
2329 011304 005010      CLR      (R0)      ;CLEAR MAP REGISTER
2330 011306 011037 001170      MOV      (R0),$TMP0      ;SEE IF MAP REGISTER IS ZERO
2331 011312 001402      BEQ      2$      ;BRANCH IF REGISTER IS ZERO
2332 011314 004737 005272      JSR      PC,WRITEERROR      ;REPORT AND LOG ERROR
2333 011320 062700 000004      2$: ADD      #4,R0      ;POINT TO NEXT MAP REG UPPER 6 BITS
2334 011324 022700 170276      CMP      #MAPH17,R0      ;SEE IF TESTING IS OVER
2335 011330 103364      BHIS     1$      ;BRANCH IF MORE REGS TO TEST
2336 011332 012737 011270 001110      MOV      #20$,$LPERR      ;INITIALIZE LOOPING POINTER IN CASE
2337                                     ;OF INTERMITTENT FAULTS.
2338 011340 005737 001254      TST      ERRCNT      ;WERE THERE ANY ERRORS ON THIS
2339 011344 001401      BEQ      TST5      ;BRANCH IF NO ERRORS
2340 011346 104011      ERROR     11      ;SUMMARY OF MAP REGS THAT COULD NOT
2341                                     ;HOLD ZERO
2342
2343
2344
```

*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:

```
2356 011350
2357 011350 000004
2358 011352 012737 011440 001324      SCOPE
2359                                     MOV      #TST6,NXTTST      ;SAVE STARTING ADDRESS OF NEXT TEST
2360 011360 012700 170300      20$: MOV      #MAPL20,R0      ;FOR ESCAPE ON PARITY ERRORS
2361 011364 012737 011372 001110      MOV      #1$,$LPERR      ;LOAD STARTING MAP REGISTER ADDR
2362 011372 000240      1$: NOP                                     ;SET LOOP ON ERROR POINTER HERE
2363 011374 005010      CLR      (R0)      ;THIS IS A SYNC POINT FOR SCOPING
2364 011376 011037 001170      MOV      (R0),$TMP0      ;CLEAR MAP REGISTER
2365 011402 001402      BEQ      2$      ;SEE IF MAP REGISTER IS ZERO
2366 011404 004737 005272      JSR      PC,WRITEERROR      ;BRANCH IF REGISTER IS ZERO
2367 011410 062700 000004      2$: ADD      #4,R0      ;REPORT AND LOG ERROR
2368 011414 022700 170374      CMP      #MAPL37,R0      ;POINT TO NEXT REGISTER
2369 011420 103364      BHIS     1$      ;SEE IF THIS SECTION IS TESTED
2370 011422 012737 011360 001110      MOV      #20$,$LPERR      ;BRANCH IF NOT
2371                                     ;INITIALIZE LOOPING POINTER IN CASE
2372 011430 005737 001254      TST      ERRCNT      ;OF INTERMITTENT FAULTS.
2373 011434 001401      BEQ      TST6      ;WERE THERE ANY ERRORS ON THIS
2374 011436 104010      ERROR     10      ;BRANCH IF NO ERRORS
2375                                     ;SUMMARY OF MAP REGS THAT COULD NOT
2376                                     ;HOLD ZERO
2377
2378
2379
```

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


```

: *
: * THIS TEST WILL ENSURE THAT MAPH20 - MAPH37 WILL ALL HOLD ZERO.
: * IF ANY REGISTERS ARE NOT ZERO AFTER BEING CLEARED, THEIR ADDRESS AND
: * CONTENTS ARE REPORTED. IF ALL THE REGISTERS HAVE RANDOM DATA THEN
: * PROBABLY NO WRITE PULSE IS BEING GENERATED, BUT IF THE ERROR IS IN
: * ONLY ONE REGISTER OR IN ONE BIT IT MIGHT BE MORE DIFFICULT TO FIND.
: * IT IS POSSIBLE THAT THE COUNT PATTERN TEST FOR MAPH20 - MAPH37 WILL
: * GIVE MORE HELP.
: *
: * *****
TST6:
SCOPE
MOV #TST7,NXTTST ;SAVE STARTING ADDRESS OF NEXT TEST
;FOR ESCAPE ON PARITY ERRORS
20$: MOV #MAPH20,R0 ;LOAD STARTING MAP REGISTER ADDR
MOV #1$,$LPERR ;SET LOOP ON ERROR POINTER HERE
1$: 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$: ADD #4,R0 ;POINT TO NEXT HIGH REGISTER
CMP #MAPH37,R0 ;SEE IF THERE ARE MORE REGS LEFT
BHS 1$ ;BRANCH IF MORE REGS TO TEST
MOV #20$,$LPERR ;INITIALIZE LOOPING POINTER IN CASE
;OF INTERMITTENT FAULTS.
TST ERRCNT ;WERE THERE ANY ERRORS ON THIS TEST
BEQ TST7 ;:BRANCH IF NO ERRORS
ERROR 11 ;SUMMARY OF MAP REGS THAT COULD NOT
;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
: *
: * THIS TEST WILL ENSURE THAT THE DATA PATH TO AND FROM THE UNIBUS
: * MAP REGISTERS IS FUNCTIONING PROPERLY. IT RUNS A COUNT PATTERN
: * THROUGH MAPL00, AND IF IT DETECTS AN ERROR THE EXPECTED COUNT
: * AND THE RECEIVED COUNT ARE REPORTED. THE TEST THEN TRIES TO RUN
: * THE COUNT THROUGH MAPL20, IN CASE THE ERROR IS IN THE LOWER
: * GROUP OF REGISTERS AND NOT IN THE DATA PATH.
: * THE DATA INPUT TO THE MAP REGISTERS IS RECEIVED FROM THE UNIBUS
: * ON 'MAPA' AND THE OUTPUT TO THE UNIBUS IS DRIVEN ON 'MAPJ'.
: *
: * *****
TST7:
SCOPE
MOV #TST10,NXTTST ;SAVE STARTING ADDRESS OF NEXT TEST

```

Address	Hex	Hex	Hex	Hex	Label	Instruction	Comment
2436	011540	012737	000012	001204		MOV	#12,\$TIMES
2437	011546	005003			20\$:	CLR	R3
2438	011550	005002				CLR	R2
2439	011552	012737	011560	001110		MOV	#1\$,\$LPERR
2440	011560	000240			1\$:	NOP	
2441	011562	010237	170200			MOV	R2,MAPL00
2442	011566	013700	170200			MOV	MAPL00,R0
2443	011572	020200				CMP	R2,R0
2444	011574	001402				BEQ	2\$
2445	011576	104012				ERROR	12
2446	011600	000404				BR	3\$
2447							
2448							
2449	011602	062702	000002		2\$:	ADD	#2,R2
2450	011606	001364				BNE	1\$
2451	011610	000416				BR	10\$
2452	011612	012737	011620	001110	3\$:	MOV	#12\$,\$LPERR
2453	011620	000240			12\$:	NOP	
2454	011622	010337	170300			MOV	R3,MAPL20
2455	011626	013701	170300			MOV	MAPL20,R1
2456	011632	020301				CMP	R3,R1
2457	011634	001003				BNE	4\$
2458	011636	062703	000002			ADD	#2,R3
2459	011642	001357				BNE	2\$
2460	011644	104013			4\$:	ERROR	13
2461	011646	012737	011546	001110	10\$:	MOV	#20\$,\$LPERR

```

*****
*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'.
*

```

2477	011654				TST10:				
2478	011654	000004				SCOPE			
2479	011656	012737	012054	001324		MOV	#TST11,NXTTST	:	SAVE STARTING ADDRESS OF NEXT TEST
2480								:	FOR ESCAPE ON PARITY ERRORS
2481	011664	012737	000002	001204		MOV	#2,\$TIMES	:	DO 2 ITERATIONS
2482	011672	005002			20\$:	CLR	R2	:	START COUNT PATTERN AT 0
2483	011674	005003				CLR	R3	:	START COUNT PATTERN AT 0
2484	011676	005004				CLR	R4	:	START COUNT AT ZERO
2485	011700	105737	001331			TSTB	KB11EM	:	IS THIS A KB11-EM?
2486	011704	001003				BNE	55\$	:	BR IF IT IS
2487	011706	105737	001332			TSTB	KB11CM	:	IS THIS A MODIFIED PROCESSOR?
2488	011712	001403				BEQ	56\$	:	NO,CARRY ON
2489	011714	012737	012052	012046	55\$:	MOV	#72\$+4,72\$;USE	:	THE MODIFIED MASK
2490	011722	012737	011730	001110	56\$:	MOV	#1\$,\$LPERR	:	SET LOOP ON ERROR POINTER TO 1\$
2491	011730	000240			1\$:	NOP		:	THIS IS A SYNC POINT FOR SCOPING


```
2492 011732 010402      MOV      R4,R2      ;MOVE COUNT TO R2
2493 011734 010237 170202  MOV      R2,@MAPH0    ;LOAD MAP REG0 HIGH 6 BITS
2494 011740 013700 170202  MOV      MAPH0,R0    ;READ BACK MAP REG 00 UPPER SIX BITS
2495 011744 047702 000076  BIC      @72$,R2 ; ;MASK OUT UNUSED BITS
2496 011750 020200      CMP      R2,R0      ;COMPARE DATA AND PATTERN LOADED
2497 011752 001402      BEQ      2$      ;BRANCH IF DATA IS CORRECT
2498 011754 104012      ERROR    12      ;POSSIBLE DATA PATH ERROR
2499 011756 000405      BR       3$      ;BRANCH TO TRY SECOND BANK OF MAP REGS
2500 011760 005204      2$: INC      R4      ;CHANGE DATA PATTERN
2501 011762 022704 177777  CMP      #177777,R4 ;HAS COUNT REACH MAXIMUM?
2502 011766 001360      BNE      1$      ;BRANCH IF TEST NOT OVER
2503 011770 000422      BR       10$     ;UPPER SIX BIT DATA PATH IS OKAY
2504 011772 012737 012000 001110 3$: MOV      #12$,$LPERR ;SET LOOP ON ERROR POINTER TO 12$
2505 012000 000240      12$: NOP      ;THIS IS A SYNC POINT FOR SCOPING
2506 012002 010403      MOV      R4,R3
2507 012004 010337 170302  MOV      R3,@MAPH20   ;LOAD MAP REG 20 HIGH 6 BITS
2508 012010 013701 170302  MOV      MAPH20,R1  ;READ BACK MAP REG 20 UPPER SIX BITS
2509 012014 047703 000026  BIC      @72$,R3
2510 012020 020301      CMP      R3,R1      ;COMPARE DATA AND PATTERN
2511 012022 001004      BNE      4$      ;BRANCH IF DATA DOES NOT MATCH
2512 012024 005204      INC      R4      ;CHANGE DATA PATTERN
2513 012026 022704 177777  CMP      #177777,R4 ;HAS COUNT REACHED MAXIMUM?
2514 012032 001352      BNE      2$      ;BRANCH IF TEST NOT OVER
2515 012034 104013      4$: ERROR    13      ;PROBABLY IS A REAL DATA PATH ERROR
2516 012036 012737 011672 001110 10$: MOV      #20$,$LPERR ;SET LOOP POINTER TO START OF TEST
2517 012044 000403      BR       72$+6
2518 012046 012050      72$: .WORD    72$+2 ;POINTER TO PROPER MASK
2519 012050 177700      .WORD    177700 ;NONMODIFIED PROCESSORS USE THIS
2520 012052 077700      .WORD    077700 ;MODIFIED PROCESSORS USE THIS
```

```
2521
2522 *****
2523 :*TEST 11      DUAL ADDRESSING ON LOADING AND READING MAP REGISTERS
2524 :*
2525 :*      THIS TEST ENSURES THAT ONLY ONE UNIBUS MAP REGISTER IS LOADED
2526 :*      DURING A 'MOV #DATA,@MAPREG' INSTRUCTION. ALL MAP REGISTERS
2527 :*      ARE CLEARED AND ONE REGISTER AT A TIME, STARTING WITH MAPLOO,
2528 :*      IS LOADED A NEGATIVE ONE. THEN, ALL MAP REGISTERS ARE READ,
2529 :*      STARTING WITH MAPH37, AND VERIFIED TO BE ZERO. ANY REGISTER
2530 :*      THAT IS NOT ZERO AND WHOSE UNIBUS ADDRESS DOES NOT MATCH THAT
2531 :*      OF THE REGISTER UNDER TEST IS REPORTED TO BE IN ERROR. AT THE
2532 :*      END OF THE TEST A SUMMARY OF ALL DUALED REGISTERS IS GIVEN.
2533 :*      THE SIGNALS THAT ARE TESTED HERE ARE ON 'MAPC' AND ARE:
2534 :*      'EN LOREG' & 'EN HIREG'. THE 'A' INPUTS TO THE ADDRESS
2535 :*      MULTIPLEXER SHOULD BE THE ONES IN USE DURING A READ OR WRITE
2536 :*      TO THE UNIBUS MAP REGISTERS.
2537 :*
2538 *****
```

```
2539 012054      TST11:
2540 012054 000004      SCOPE
2541 012056 012737 012200 001324  MOV      #TST12,NXTTST ;SAVE STARTING ADDRESS OF NEXT TEST
2542      ;FOR ESCAPE ON PARITY ERRORS
2543 012064 012737 000144 001204  MOV      #144,$TIMES ;DO 144 ITERATIONS
2544 012072 012700 170200      20$: MOV      #MAPLO,R0 ;LOAD FIRST MAP REG ADDR INTO R0
2545 012076 012737 012104 001110  MOV      #1$,$LPERR ;SET LOOP ON ERROR POINTER HERE
2546 012104 012701 170376      1$: MOV      #MAPH37,R1 ;PUT ADDRESS OF HIGHEST MAP REG IN R1
2547 012110 004737 005134      JSR      PC,CLRMAP ;CLEAR ALL MAP REGISTERS IN CASE OF
```

```
2548
2549 012114 000240      NOP
2550 012116 012710 177777  MOV    #177777,(R0)
2551 012122 011102      2$: MOV    (R1),R2
2552 012124 001404      BEQ    3$
2553 012126 020100      CMP    R1,R0
2554 012130 001402      BEQ    3$
2555 012132 004737 005356  JSR    PC,DUALADR
2556 012136 162701 000002  3$: SUB    #2,R1
2557 012142 022701 170200  CMP    #MAPLO,R1
2558 012146 101765      BLOS   2$
2559 012150 062700 000002  ADD    #2,R0
2560 012154 022700 170376  CMP    #MAPH37,R0
2561 012160 103351      BHS    1$
2562 012162 012737 012072 001110 MOV    #20$,$LPERR
2563
2564 012170 005737 001254  TST    ERRCNT
2565 012174 001401      BEQ    TST12
2566 012176 104014      ERROR  14
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
```

:A 'FLAKEY' PROBLEM
:THIS IS A SYNC POINT FOR SCOPING
:LOAD MAP REGISTER UNDER TEST
:READ MAP REGS STARTING FROM TOP
:GO TO 3\$ IF ZERO
:SEE IF NON-ZERO ADDRS MATCH
:GO TO 3\$ IF MATCH
:JUMP TO SUBROUTINE TO LOG ALL ERRORS
:POINT TO NEXT MAP REGISTER
:HAVE ALL REGISTERS BEEN READ
:IF NOT CONTINUE TEST
:POINT TO NEXT MAP REGISTER
:SEE IF TEST IS OVER
:CONTINUE TESTING
:INITIALIZE LOOPING POINTER IN CASE
:OF INTERMITTENT FAULTS.
:SEE IF THERE WERE ANY ERRORS
:GOTO NEXT TEST IF NO ERRORS
:SUMMARY OF DUAL ADDRESSING ERRORS
:ON LOADING MAP REGISTERS

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

:*****
:TEST 12 COUNT PATTERN LOW 20 MAP REGISTERS LOWER 16 BITS
:*****
:THIS TEST WILL RUN A COUNT PATTERN THROUGH UNIBUS MAP REGISTERS
:MAPLOO - MAPL17. IF THE COUNT RECEIVED DOES NOT MATCH THE
:EXPECTED PATTERN THEN THE MAP REGISTER ADDRESS, DATA RECEIVED,
:PATTERN LOADED, AND PATTERN EXPECTED ARE REPORTED. AT THE END
:OF THE TEST A SUMMARY OF ALL ERRORS IS GIVEN SO THAT YOU CAN
:DETERMINE IF THE ERROR IS BIT SENSITIVE OR REGISTER SENSITIVE.
:THE REGISTERS ARE ON 'MAPC' AND THE OUTPUT MULTIPLEXER IS ON
:'MAPJ'.
:*****

```
TST12:
SCOPE
MOV    #TST13,NXTTST  :SAVE STARTING ADDRESS OF NEXT TEST
:FOR ESCAPE ON PARITY ERRORS
:DO 12 ITERATIONS
20$: MOV    #12,$TIMES
CLR    R2
:START WITH AN ALL ZERO PATTERN
MOV    #2$,$LPERR
:SET LOOP ON ERROR POINTER TO 2$
1$: MOV    R2,R4
:PUT COUNT IN R4 FOR MASK
BIC    #000001,R4
:CLEAR BITS IN MASK FOR PROPER COMPARE
MOV    #MAPLOO,R0
:LOAD STARTING MAP REGISTER ADDRESS
2$: NOP
MOV    R2,(R0)
:THIS IS A SYNC POINT FOR SCOPING
MOV    (R0),R3
:LOAD MAP REGISTER WITH COUNT PATTERN
CMP    R4,R3
:READ MAP REGISTER INTO R3
BEQ    3$
:COMPARE MASK WITH DATA
: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    #MAPL17,R0
:SEE IF ALL REGISTERS HAVE BEEN LOADED
```


2615
2616
2617
2618
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

IST13:

```
TST13:      SCOPE
MOV          #TST14,NXTTST    ;SAVE STARTING ADDRESS OF NEXT TEST
                                ;FOR ESCAPE ON PARITY ERRORS
20$:        MOV          #144,$TIMES    ;;DO 144 ITERATIONS
CLR          R2                ;START WITH AN ALL ZERO PATTERN
1$:         MOV          #2$,$LPERR     ;SET LOOP ON ERROR POINTER TO 2$
MOV          R2,R4             ;PUT COUNT IN R4 FOR MASK
BIC          #177700,R4        ;CLEAR BITS IN MASK FOR PROPER COMPARE
2$:         MOV          #MAPH00,R0     ;LOAD STARTING MAP REGISTER ADDRESS
NOP                      ;THIS IS A SYNC POINT FOR SCOPING
MOV          R2,(R0)           ;LOAD MAP REGISTER WITH COUNT PATTERN
MOV          (R0),R3           ;READ MAP REGISTER INTO R3
CMP          R4,R3             ;COMPARE MASK WITH DATA
BEQ          3$               ;BRANCH IF DATA MATCHES
JSR          PC,COUNT          ;JUMP TO COUNT TO LOG ALL ERRORS
3$:         CLR          R3          ;CLEAR DATA HOLDER
ADD          #4,R0             ;POINT TO NEXT MAP REGISTER UNDER TEST
CMP          #MAPH17,R0        ;SEE IF ALL REGISTERS HAVE BEEN LOADED
BHIS        2$               ;BRANCH IF STILL MORE TO GO
CMP          #177,R2 ;SEE IF   COUNT HAS CYCLED
BEQ          4$               ;BRANCH IF TEST IS DONE
ADD          #1,R2            ;CHANGE COUNT PATTERN
BR          1$                 ;CONTINUE TESTING
4$:         MOV          #20$,$LPERR    ;INITIALIZE LOOPING POINTER IN CASE
                                ;OF INTERMITTENT FAULTS.
TST          ERRCNT            ;SEE IF THERE WERE ANY ERRORS
BEQ          TST14             ;;BRANCH IF NO ERRORS
ERROR       16                 ;SUMMARY OF COUNT PATTERN FAILURES
                                ;IN MAPPING REGISTERS
```

2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715

```

*****
*TEST 14      COUNT PATTERN HIGH 20 MAP REGISTERS LOWER 16 BITS
*
*      THIS TEST WILL RUN A COUNT PATTERN THROUGH UNIBUS MAP REGISTERS
*      MAPL20 - MAPL37.  IF THE COUNT RECIEVED DOES NOT MATCH THE
*      EXPECTED PATTERN THEN THE MAP REGISTER ADDRESS, DATA RECEIVED,
*      PATTERN LOADED, AND PATTERN EXPECTED ARE REPORTED.  AT THE END
*      OF THE TEST A SUMMARY OF ALL ERRORS IS GIVEN SO THAT YOU CAN
*      DETERMINE IF THE ERROR IS BIT SENSITIVE OR REGISTER SENSITIVE.
*      THE REGISTERS ARE ON 'MAPD' AND THE OUTPUT MULTIPLEXER IS ON
*      'MAPJ'.
*****
TST14:
SCOPE
MOV      #TST15,NXTTST      ;SAVE STARTING ADDRESS OF NEXT TEST
;FOR ESCAPE ON PARITY ERRORS
;DO 12 ITERATIONS
MOV      #12,$TIMES
CLR      R2
;START WITH AN ALL ZERO PATTERN
MOV      #2,$LPERR
;SET LOOP ON ERROR POINTER TO 2$
;PUT COUNT IN R4 FOR MASK
MOV      R2,R4
BIC      #000001,R4
;CLEAR BITS IN MASK FOR PROPER COMPARE
MOV      #MAPL20,R0
;LOAD STARTING MAP REGISTER ADDRESS
;THIS IS A SYNC POINT FOR SCOPING
NOP
;LOAD MAP REGISTER WITH COUNT PATTERN
MOV      R2,(R0)
MOV      (R0),R3
;READ MAP REGISTER INTO R3
CMP      R4,R3
;COMPARE MASK WITH DATA
BEQ      3$
;BRANCH IF DATA MATCHES
JSR      PC,COUNT
;JUMP TO COUNT TO LOG ALL ERRORS
CLR      R3
;CLEAR DATA HOLDER
;POINT TO NEXT MAP REGISTER UNDER TEST
ADD      #4,R0
;SEE IF ALL REGISTERS HAVE BEEN LOADED
CMP      #MAPL37,R0
;BRANCH IF STILL MORE TO GO
BHS      2$
;SEE IF COUNT HAS CYCLED
CMP      #177777,R2
;BRANCH IF TEST IS DONE
BEQ      4$
;CHANGE COUNT PATTERN
ADD      #401,R2
;CONTINUE TESTING
BR       1$
;INITIALIZE LOOPING POINTER IN CASE
;OF INTERMITTENT FAULTS.
MOV      #20,$LPERR
;SEE IF THERE WERE ANY ERRORS
TST      ERRCNT
;BRANCH IF NO ERRORS
BEQ      TST15
;SUMMARY OF COUNT PATTERN FAILURES
;IN MAPPING REGISTERS
ERROR    15

```

```

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

```



```

2716      ;*      'MAPJ'.
2717      ;*****
2718      TST15:
2719      SCOPE
2720      MOV      #TST16,NXTTST      ;SAVE STARTING ADDRESS OF NEXT TEST
2721      ;FOR ESCAPE ON PARITY ERRORS
2722      MOV      #144,$TIMES      ;DO 144 ITERATIONS
2723      CLR      R2      ;START WITH AN ALL ZERO PATTERN
2724      MOV      #2,$$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
2725      MOV      R2,R4      ;PUT COUNT IN R4 FOR MASK
2726      BIC      #177700,R4      ;CLEAR BITS IN MASK FOR PROPER COMPARE
2727      MOV      #MAPH20,R0      ;LOAD STARTING MAP REGISTER ADDRESS
2728      NOP      ;THIS IS A SYNC POINT FOR SCOPING
2729      MOV      R2,(R0)      ;LOAD MAP REGISTER WITH COUNT PATTERN
2730      MOV      (R0),R3      ;READ MAP REGISTER INTO R3
2731      CMP      R4,R3      ;COMPARE MASK WITH DATA
2732      BEQ      3$      ;BRANCH IF DATA MATCHES
2733      JSR      PC,COUNT      ;JUMP TO COUNT TO LOG ALL ERRORS
2734      CLR      R3      ;CLEAR DATA HOLDER
2735      ADD      #4,R0      ;POINT TO NEXT MAP REGISTER UNDER TEST
2736      CMP      #MAPH37,R0      ;SEE IF ALL REGISTERS HAVE BEEN LOADED
2737      BHS      2$      ;BRANCH IF STILL MORE TO GO
2738      CMP      #177,R2 ;SEE IF COUNT HAS CYCLED
2739      BEQ      4$      ;BRANCH IF TEST IS DONE
2740      ADD      #1,R2      ;CHANGE COUNT PATTERN
2741      BR      1$      ;CONTINUE TESTING
2742      MOV      #20,$$LPERR      ;INITIALIZE LOOPING POINTER IN CASE
2743      ;OF INTERMITTENT FAULTS.
2744      TST      ERRCNT      ;SEE IF THERE WERE ANY ERRORS
2745      BEQ      TST16      ;BRANCH IF NO ERRORS
2746      ERROR      16      ;SUMMARY OF COUNT PATTERN FAILURES
2747      ;IN MAPPING REGISTERS
2748
2749
2750      ;*****
2751      ;*TEST 16      CACHE REGISTER DATA PATHS
2752      ;*
2753      ;*      THIS TEST MAKES AN EFFORT TO VERIFY THE DATA PATH FROM THE
2754      ;*      CACHE REGISTERS THROUGH THE UNIBUS MAP TO THE CPU. IT FIRST
2755      ;*      CHECKS THE CONTROL REGISTER (177746) TO SEE IF THE CACHE IS
2756      ;*      DISABLED, IF IT IS NOT DISABLED THEN THE CONTROL REGISTER
2757      ;*      IS CLEARED AND VERIFIED TO BE ZERO. THE MAINTENANCE REGISTER
2758      ;*      IS ALWAYS CLEARED AND VERIFIED TO BE ZERO. THEN THE CACHE
2759      ;*      ADDRESS REGISTERS (177740 & 177742) ARE READ, WITH THE ERROR
2760      ;*      REGISTER CLEARED, THEY SHOULD BE 177740 AND 000003 RESPECTIVELY.
2761      ;*      IF ANY OF THESE CONDITIONS ARE NOT MET AN ERROR IS REPORTED
2762      ;*      AS BEING A POSSIBLE CACHE REGISTER DATA PATH ERROR.
2763      ;*      THE INPUT TO THE CACHE REGISTERS IS FROM 'MAPA' AND THE CACHE
2764      ;*      DATA MULTIPLEXER IS ON 'MAPH'. THE REGISTER INPUTS ARE THE
2765      ;*      'X' INPPUTS, THESE GO TO 'MAPJ' TO GET BACK TO THE CPU.
2766      ;*
2767      ;*****
2768      TST16:
2769      SCOPE
2770      MOV      #TST17,NXTTST      ;SAVE STARTING ADDRESS OF NEXT TEST
2771      ;FOR ESCAPE ON PARITY ERRORS

```

```
2772 012730 012737 012750 001110 20$: MOV #10$,$LPERR ;SET LOOP ON ERROR POINTER TO 10$
2773 012736 005000 CLR R0 ;CLEAR R0 IN CASE CACHE IS DISABLED
2774 012740 032737 000014 177746 BIT #14,@#CONTRL ;SEE IF EITHER GROUP IS DISABLED
2775 012746 001012 BNE 1$ ;BRANCH IF CACHE IS DISABLED
2776 012750 000240 10$: NOP ;THIS IS A SYNC POINT FOR SCOPING
2777 012752 005005 CLR R5
2778 012754 005037 177746 CLR @#CONTRL ;CLEAR CONTROL REGISTER
2779 012760 013700 177746 21$: MOV @#CONTRL,R0 ;READ CONTROL REGISTER
2780 012764 001403 BEQ 1$ ;BRANCH IF CONTROL REGISTER IS ZERO
2781 012766 005205 INC R5 ;WAIT FOR VCIP BIT IN CCR TO
2782 012770 001373 BNE 21$ ;CLEAR
2783 012772 104017 ERROR 17 ;POSSIBLE FAULT IN CACHE REG DATA PATHS
2784 012774 012737 013002 001110 1$: MOV #11$,$LPERR ;SET LOOP ON ERROR POINTER TO 11$
2785 013002 000240 11$: NOP ;THIS IS A SYNC POINT FOR SCOPING
2786 013004 005037 177750 CLR @#MAINT ;CLEAR MAINTENANCE REGISTER
2787 013010 013701 177750 MOV @#MAINT,R1 ;READ MAINTENANCE REGISTER
2788 013014 001401 BEQ 2$ ;BRANCH IF MAINTENANCE REGISTER IS ZERO
2789 013016 104020 ERROR 20 ;MAINTENANCE REGISTER WILL NOT CLEAR
2790 013020 012737 013026 001110 2$: MOV #12$,$LPERR ;SET LOOP ON ERROR POINTER TO 12$
2791 013026 012737 177777 177744 12$: MOV #-1, MEMERR ;MAKE SURE THAT CACHE ERROR REG IS CLEAR
2792 013034 000240 -- NOP ;THIS IS A SYNC POINT FOR SCOPING
2793 013036 013700 177740 MOV LOADRS,R0 ;READ CACHE LOW ADDR REG TO R0
2794 013042 022700 177740 CMP #177740,R0 ;SEE IF R0 = ADDR OF CACHE LOW ADDR REG
2795 013046 001401 BEQ 3$ ;BRANCH IF DATA READ CORRECTLY
2796 013050 104021 ERROR 21 ;CANNOT READ 177740 FROM 'LOADRS'
2797 013052 012737 013060 001110 3$: MOV #13$,$LPERR ;SET LOOP ON ERROR POINTER TO 13$
2798 013060 012737 177777 177744 13$: MOV #-1, MEMERR ;MAKE SURE THAT CACHE ERROR REG IS CLEAR
2799 013066 000240 NOP ;THIS IS A SYNC POINT FOR SCOPING
2800 013070 013700 177742 MOV HIADRS,R0 ;READ CACHE HIGH ADDR REG TO R0
2801 013074 042700 177700 BIC #177700,R0 ;MASK OFF UPPER 10 BITS
2802 013100 022700 000003 CMP #3,R0 ;SEE IF R0 = 3
2803 013104 001401 BEQ 14$ ;BRANCH IF DATA MATCHES
2804 013106 104022 ERROR 22 ;CANNOT READ 000003 FROM 'HIADRS'
2805 013110 012737 012730 001110 14$: MOV #20$,$LPERR ;SET LOOP POINTER TO START OF TEST
```

2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827 013116

```
*****
*TEST 17 CACHE REGISTER DATA PATH COUNT PATTERN
*
* THIS TEST FIRST SAVES THE CONDITION OF THE CACHE CONTROL
* REGISTER IS '$TMP5' SO THAT THE CACHE CAN BE RESTORED.
* IF EITHER BIT 3 OR 4 IS ON THE TEST BRANCHES TO '3$' TO
* TEST BITS 5 - 15 OF THE DATA PATH, IF NEITHER BIT IS ON THEN
* THE TEST RUNS A COUNT PATTERN THRU THE CONTROL REGISTER FROM
* 00 - 77. IF ANY BITS ARE DETECTED AS BEING STUCK THE ERROR
* IS REPORTED AND A SUMMARY OF THOSE ERRORS IS GIVEN. THEN THE
* TEST FORCES MISSES IN BOTH GROUPS OF THE CACHE AND PROCEEDS
* TO RUN A COUNT PATTERN THROUGH THE CACHE MAINTENANCE REGISTER
* (177746) FROM 000000 - 177760 IN INCREMENTS OF 20.
* THE CODE IS SET UP IN SUCH A WAY THAT NO PARITY ERRORS
* SHOULD BE GENERATED. AGAIN ALL BAD MATCHES OF DATA WILL BE
* REPORTED AND A SUMMARY OF ALL ERRORS IS GIVEN AT THE END OF
* THE TEST AND THE CACHE IS RESTORED TO ITS PREVIOUS CONDITION.
*****
TST17:
```


2828	013116	000004			SCOPE		
2829	013120	012737	013540	001324	MOV	#TST20,NXTTST	:SAVE STARTING ADDRESS OF NEXT TEST
2830							:FOR ESCAPE ON PARITY ERRORS
2831	013126	012737	000012	001204	MOV	#12,\$TIMES	:DO 12 ITERATIONS
2832	013134	104416			TBITO		:TURN OFF T BIT TRAPPING IF ON
2833	013136	013737	177746	001202	20\$: MOV	@#CONTRL,\$TMP5	:SAVE CONTROL REG TO RESTORE CACHE
2834	013144	032737	000014	177746	BIT	#14,@#CONTRL	:SEE IF EITHER GROUP IS DISABLED
2835	013152	001047			BNE	3\$	:SKIP FIRST PART OF TEST IF DISABLED
2836	013154	012700	177746		MOV	#CONTRL,R0	:LOAD ADDR OF CONTROL REG INTO R0
2837	013160	005002			CLR	R2	:START COUNT PATTERN AT ZERO
2838	013162	012737	013170	001110	MOV	#1\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 1\$
2839	013170	000240			1\$: NOP		:THIS IS A SYNC POINT FOR SCOPING
2840	013172	010210			MOV	R2,(R0)	:LOAD COUNT INTO CONTROL REG
2841	013174	011003			MOV	(R0),R3	:READ COUNT FROM CONTROL REG
2842	013176	020203			CMP	R2,R3	:SEE IF COUNT = DATA READ
2843	013200	001402			BEQ	2\$	:BRANCH IF DATA MATCHES
2844	013202	004737	005576		JSR	PC,CACOUNT	:LOG AND REPORT COUNT PATTERN ERROR
2845	013206	005202			2\$: INC	R2	:CHANGE COUNT PATTERN
2846	013210	022702	000077		CMP	#77,R2	:SEE IF COUNT HAS PASSED MAXIMUM
2847	013214	103365			BHIS	1\$	:BRANCH IF COUNT HASN'T CYCLED
2848	013216	005737	001254		TST	ERRCNT	:SEE IF ANY ERRORS ON THIS CYCLE
2849	013222	001423			BEQ	3\$	:BRANCH IF NO ERRORS
2850	013224	104023			ERROR	23	:COUNT THRU CONTROL REG FAILED
2851	013226	005037	001322		CLR	RETRY	:CLEAR RETRY FLAG AN THE START OF
2852							:EACH TEST
2853	013232	005037	001254		CLR	ERRCNT	:CLEAR THE MULTIPLE ERROR COUNTER
2854	013236	005037	001232		CLR	DATADR	:LOCATION FOR LOGICAL OR OF BAD DATA
2855	013242	005037	001226		CLR	ADDRDR	:LOCATION FOR LOGICAL OR OF ADDRESS
2856	013246	005037	001236		CLR	PATTOR	:LOCATION FOR LOGICAL OR OF PATTERN LOADED
2857	013252	012700	177777		MOV	#-1,R0	:LOAD -1 INTO R0 TO INITIALIZE LOGICAL AND LOCs
2858	013256	010037	001230		MOV	R0,DATAND	:LOCATION FOR LOGICAL AND OF BAD DATA
2859	013262	010037	001224		MOV	R0,ADRAND	:LOCATION FOR LOGICAL AND OF ADDRESS
2860	013266	010037	001234		MOV	R0,PATAND	:LOCATION FOR LOGICAL AND OF PATTERN LOADED
2861	013272	012737	000014	177746	3\$: MOV	#14,@#CONTRL	:DISABLE BOTH GROUPS OF CACHE
2862	013300	012700	177750		MOV	#MAINT,R0	:LOAD ADDR OF MAINTENANCE REG INTO R0
2863	013304	010001			MOV	R0,R1	:R1 HAS #MAINT SO CLR (R1) HAS P.B.'S ON
2864	013306	005002			CLR	R2	:START COUNT PATTERN AT ZERO
2865	013310	012737	013316	001110	MOV	#4\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 4\$
2866	013316	010205			4\$: MOV	R2,R5	:PUT COUNT IN R5 FOR CORRECT PARITY
2867	013320	000240			NOP		:THIS IS A SYNC POINT FOR SCOPING
2868	013322	010510			MOV	R5,(R0)	:LOAD MAINT REGISTER INST HAS P.B.'S ON
2869	013324	011003			MOV	(R0),R3	:READ MAINT REG (P.B.'S ON)
2870	013326	005011			CLR	(R1)	:TURN OFF MAINT REG (P.B.'S ON)
2871	013330	050000			BIS	R0,R0	:DUMMY INST WITH P.B.'S ON
2872	013332	020203			CMP	R2,R3	:SEE IF COUNT = DATA READ
2873	013334	001402			BEQ	5\$	:BRANCH IF DATA MATCHES
2874	013336	004737	005576		JSR	PC,CACOUNT	:LOG AND REPORT COUNT PATTERN ERROR
2875	013342	062702	000020		5\$: ADD	#20,R2	:CHANGE COUNT PATTERN
2876	013346	001363			BNE	4\$	:BRANCH IF COUNT HAS NOT PASSED 177760
2877	013350	005737	001254		TST	ERRCNT	:SEE IF ANY ERRORS ON THIS COUNT CYCLE
2878	013354	001401			BEQ	19\$	:BRANCH IF NO ERRORS ON THIS PART
2879	013356	104024			ERROR	24	:SUMMARY OF COUNT ERRORS IN MAINT REG
2880	013360	012737	013136	001110	19\$: MOV	#20\$, \$LPERR	:SET LOOP POINTER TO START OF TEST
2881	013366	013737	001202	177746	MOV	\$TMP5,@#CONTRL	:RETURN CACHE TO PREVIOUS STATE
2882	013374	000004					

```
2884
2885
2886
2887
2888
2889 013376 104420
2890
2891 013400 012700 077406
2892
2893 013404 010037 172300
2894 013410 010037 172302
2895 013414 010037 172304
2896 013420 010037 172306
2897 013424 010037 172310
2898 013430 010037 172312
2899 013434 010037 172314
2900 013440 010037 172316
2901 013444 012737 000000 172340
2902 013452 012737 000200 172342
2903 013460 012737 000400 172344
2904 013466 012737 000600 172346
2905 013474 012737 001000 172350
2906 013502 012737 001200 172352
2907 013510 012737 170000 172354
2908 013516 012737 177600 172356
2909 013524 012737 000001 177572
2910 013532 012737 000020 172516
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923 013540
2924 013540 000004
2925 013542 012737 013572 001106
2926 013550 012737 013572 001110
2927 013556 012737 000020 001102
2928 013564 013737 001102 177570
2929
2930 013572 012737 000020 001264 20$:
2931 013600 012737 013640 001110
2932 013606 012737 177702 172354
2933 013614 012702 175254
2934 013620 010237 170200
2935 013624 012700 004000
2936 013630 012704 140000
2937 013634 074037 172354 1$:
2938 013640 000240 10$:
2939 013642 011401
```

```

: * 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).

TBITR ;RESTORE THE T BIT TO ITS CONDITION
;BEFORE THE LAST TEST
MOV #77406,R0 ;MAKE THE KERNEL I-SPACE PAGES ALL
;4K, UPWARD EXPANDABLE, READ/WRITE
MOV R0,KIPDR0 ;KERNEL I-SPACE PAGE 0
MOV R0,KIPDR1 ;KERNEL I-SPACE PAGE 1
MOV R0,KIPDR2 ;KERNEL I-SPACE PAGE 2
MOV R0,KIPDR3 ;KERNEL I-SPACE PAGE 3
MOV R0,KIPDR4 ;KERNEL I-SPACE PAGE 4
MOV R0,KIPDR5 ;KERNEL I-SPACE PAGE 5
MOV R0,KIPDR6 ;KERNEL I-SPACE PAGE 6
MOV R0,KIPDR7 ;KERNEL I-SPACE PAGE 7
MOV #000,KIPAR0 ;MAP TO PHYSICAL 0
MOV #200,KIPAR1 ;MAP TO PHYSICAL 4K - 8K
MOV #400,KIPAR2 ;MAP TO PHYSICAL 8K - 12K
MOV #600,KIPAR3 ;MAP TO PHYSICAL 12K - 16K
MOV #1000,KIPAR4 ;MAP TO PHYSICAL 16K - 20K
MOV #1200,KIPAR5 ;MAP TO PHYSICAL 20K - 24K
MOV #170000,KIPAR6 ;MAP TO UNIBUS
MOV #177600,KIPAR7 ;MAP TO I/O PAGE
MOV #BIT0,MMR0 ;ENABLE FULL 18-BIT MAPPING
MOV #BIT4,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 MAPL00, THRU KIPAR6. THE TEST THEN CHANGES ONE BIT
: * AT A TIME IN PAR6 SO THAT IT SHOULD NEVER REFERENCE MAPL00. IF
: * IT DOES AN ERROR IS REPORTED.
: *****
TST20:
SCOPE
MOV #20$,SLPADR ;SET LOOP ON TEST POINTER TO 20$
MOV #20$,SLPERR ;SET LOOP ON ERROR POINTER TO 20$
MOV #20,$TSTNM ;SETUP TEST NUMBER AND CLR ERROR FLAG
MOV $TSTNM,DISPLAY ;DISPLAY TEST NUMBER FOR ALL TO SEE
.EQUIV BIT4,TIMOUT ;BIT4 IS TIMEOUT BIT IN CPU ERROR REG
20$: MOV #TIMOUT,CPUEXP ;EXPECTING CPU TIME OUT ON UNIBUS
MOV #10$,SLPERR ;SET LOOP ON ERROR POINTER TO 10$
MOV #177702,KIPAR6 ;PUT MAP REG 0 ADDR IN PAR6
MOV #175254,R2 ;PATTERN FOR TESTING.
MOV R2,@MAPL0 ;LOAD MAP REGISTER 0
MOV #BIT11,R0 ;SET BIT 11 TO FLOAT THRU PAR6
MOV #140000,R4 ;VIRT.ADDR. TO SELECT PAR6
1$: XOR R0,KIPAR6 ;CHANGE A BIT OF MAP REG 0'S ADDR
10$: NOP ;THIS IS A SYNC POINT FOR SCOPING
MOV (R4),R1 ;READ LOCATION POINTED TO BY PAR6
```



```
2940 013644 020201      CMP      R2,R1      ;SEE IF DATA FETCHED MATCHES PATTERN
2941 013646 001007      BNE      3$      ;BRANCH IF NOT SAME
2942 013650 012714 000000 MOV      #0,(R4)      ;TRY TO CLEAR THIS LOCATION
2943 013654 005737 170200 TST      @MAPLO      ;SEE IF MAP REG 0 GOT CLEARED
2944 013660 001001      BNE      2$      ;BRANCH IF MAP REG NOT ZERO
2945 013662 104025      ERROR    25      ;GOT TO MAPLO WITH ONE BIT
2946                                ;DIFFERENT IN ADDRESS FROM 770200
2947 013664 010114      2$: MOV      R1,(R4)      ;RELOAD THE LOCATION
2948 013666 074037 172354 3$: XOR      R0,KIPAR6      ;RESTORE BIT TO ORIGINAL STATUS
2949 013672 006200      ASR      R0      ;RIGHT SHIFT ONE PLACE
2950 013674 020027 000001 CMP      R0,#1      ;SEE IF TEST IS OVER
2951 013700 001355      BNE      1$      ;BRANCH TO CONTINUE TEST
2952 013702 012737 013572 001110 MOV      #20$,$LPERR      ;SET LOOP POINTER TO START OF TEST
2953 013710 005037 001264 CLR      CPUEXP      ;ZERO EXPECTED CPU TRAP FLAG
2954
2955 *****
2956 *TEST 21      CACHE REGISTER ADDRESS DECODE TEST
2957 *
2958 *      THIS TEST TRIES TO VERIFY THAT NONE OF THE INPUTS TO THE ADDRESS
2959 *      DECODER FOR THE CACHE REGISTERS ON 'MAPB' IS STUCK TRUE.
2960 *      KIPAR6 IS SETUP TO HOLD 177740 AND R4 HAS THE VIRTUAL ADDRESS
2961 *      TO SELECT 'LOADRS', THRU KIPAR6. THE TEST THEN CHANGES ONE BIT
2962 *      AT A TIME IN PAR6 SO THAT IT SHOULD NEVER REFERENCE 'LOADRS'.
2963 *      IF IT DOES AN ERROR IS REPORTED.
2964 *
2965 *****
2966 TST21:
2967 013714 000004      SCOPE
2968 013716 012737 014116 001324 MOV      #TST22,NXTTST      ;SAVE STARTING ADDRESS OF NEXT TEST
2969                                ;FOR ESCAPE ON PARITY ERRORS
2970 013724 012737 000020 001264 20$: MOV      #TIMOUT,CPUEXP      ;EXPECTING UNIBUS TIMEOUT IN THIS TEST
2971 013732 012737 013766 001110 MOV      #10$,$LPERR      ;SET LOOP ON ERROR POINTER TO 10$
2972 013740 012737 177777 172354 MOV      #177777,KIPAR6      ;PUT CACHE REG BASE IN PAR6
2973 013746 012702 177740      MOV      #177740,R2      ;PUT DATA PATTERN IN R2
2974 013752 012700 000040      MOV      #BIT5,R0      ;SET BIT 5 TO FLOAT THRU PAR6
2975 013756 012704 140040      MOV      #140040,R4      ;VIRT. ADDR. TO SELECT PAR6
2976 013762 074037 172354      1$: XOR      R0,KIPAR6      ;CHANGE A BIT OF CACHE REG'S ADDR
2977 013766 000240      10$: NOP
2978 013770 011401      MOV      (R4),R1      ;THIS IS A SYNC POINT FOR SCOPING
2979                                ;TRY TO READ CACHE 'LOADRS' WITH
2980                                ;ONE BIT DIFFERENT THAN ADDR. 777740
2980 013772 020201      CMP      R2,R1      ;SEE IF YOU GOT CACHE 'LOADRS'
2981 013774 001001      BNE      2$      ;BRANCH TO CONTINUE TEST.
2982 013776 104026      ERROR    26      ;READ CACHE 'LOADRS' WITHOUT ADRS 777740
2983 014000 074037 172354      2$: XOR      R0,KIPAR6      ;RESTORE BIT TO ORIGINAL STATUS.
2984 014004 006200      ASR      R0      ;RIGHT SHIFT R0 1 PLACE
2985 014006 001365      BNE      1$      ;BRANCH IF NOT 0
2986 014010 012737 014022 001110 MOV      #11$,$LPERR      ;SET LOOP ON ERROR POINTER TO 11$
2987 014016 012701 177700      MOV      #177700,R1      ;PUT ADDRESS IN R1
2988 014022 000240      11$: NOP      ;THIS IS A SYNC POINT FOR SCOPING
2989 014024 011103      MOV      (R1),R3      ;TRY TO READ 177700
2990 014026 020203      CMP      R2,R3      ;SEE IF YOU GET 177740
2991 014030 001001      BNE      3$      ;BRANCH IF YOU DON'T
2992 014032 104027      ERROR    27      ;READ CACHE 'LOADRS' WITHOUT ADRS 777740
2993 014034 012737 014046 001110 3$: MOV      #12$,$LPERR      ;SET LOOP ON ERROR POINTER TO 12$
2994 014042 012701 177760      MOV      #177760,R1      ;PUT ADDRESS IN R1
2995 014046 000240      12$: NOP      ;THIS IS A SYNC POINT FOR SCOPING
```

```
2996 014050 011103      MOV      (R1),R3      ;TRY TO READ 177760
2997 014052 020203      CMP      R2,R3      ;SEE IF YOU GET 177740
2998 014054 001001      BNE      4$      ;BRANCH IF YOU DON'T
2999 014056 104027      ERROR    27      ;READ CACHE 'LOADRS' WITHOUT ADRS 777740
3000 014060 012737 014072 001110 4$:      MOV      #13$,$LPERR      ;SET LOOP ON ERROR POINTER TO 13$
3001 014066 012701 177754      MOV      #177754,R1      ;PUT ADDRESS IN R1
3002 014072 000240      NOP              ;THIS IS A SYNC POINT FOR SCOPING
3003 014074 011103      MOV      (R1),R3      ;TRY TO READ 177754
3004 014076 020203      CMP      R2,R3      ;SEE IF YOU GET 177740
3005 014100 001001      BNE      19$      ;BRANCH IF YOU DIDN'T READ ADRS 177740
3006 014102 104027      ERROR    27      ;READ CACHE 'LOADRS' WITHOUT ADRS 777740
3007 014104 012737 013724 001110 19$:      MOV      #20$,$LPERR      ;SET LOOP POINTER TO START OF TEST
3008 014112 005037 001264      CLR      CPUEXP      ;ZERO EXPECTED CPU TRAP CONDITION
3009
3010
3011
3012
3013
3014
3015
3016
3017
3018
3019
3020
3021
3022
3023
3024
3025
3026
3027
3028 014116
3029 014116 000004
3030 014120 012737 014350 001324
3031
3032 014126 012737 000012 001204
3033 014134 004737 005134
3034 014140 012737 000020 001264
3035 014146 012737 014154 001110
3036 014154 012737 170400 172354
3037 014162 005037 001266
3038 014166 000240
3039 014170 013700 140000
3040 014174 005737 001266
3041 014200 001412
3042 014202 062737 000200 172354
3043 014210 022737 177600 172354
3044 014216 001361
3045 014220 104030
3046 014222 000137 014500
3047 014226 013737 172354 172352
3048 014234 042737 170000 172352
3049 014242 012737 173214 120000
3050 014250 013701 140000
3051 014254 022701 173214

:*****
:TEST 22      DATA PATH, UNIBUS TO MAIN MEMORY
:
:      THIS TEST RUNS A COUNT PATTERN THROUGH A MEMORY LOCATION VIA
:      THE UNIBUS.  THE UNIBUS MAP IS LEFT OFF DURING THIS TEST SO
:      THAT THE ADDRESS IS NOT RELOCATED.  THE TEST TRIES TO LOAD THE
:      PATTERN INTO ADDRESS 040000 (8K) BUT IF THE MAP JUMPERS ARE
:      SET NOT TO RESPOND TO THAT ADDRESS THE NEXT 4K IS TRIED UNTIL
:      THE TEST GETS TO MAIN MEMORY FROM THE UNIBUS.  IF THIS TEST
:      DETERMINES THAT IT CANNOT GET TO MAIN MEMORY FROM THE UNIBUS
:      IT REPORTS THE FACT AND SKIPS THE NEXT TEST FOR VERIFICATION.
:      THE DATA PATH TO MAIN MEMORY FROM THE UNIBUS IS ON 'MAPA' AND
:      THE PATH FROM MAIN MEMORY TO THE UNIBUS IS FROM THE CACHE
:      'DTML CDMX D00' - 'DTML CDMX D15' INTO 'MAPH' THE 'Z' INPUTS
:      TO THE MULTIPLEXER AND THEN TO 'MAPJ' AND OUT TO THE UNIBUS.
:*****
TST22:
SCOPE
MOV      #TST23,NXTTST      ;SAVE STARTING ADDRESS OF NEXT TEST
:FOR ESCAPE ON PARITY ERRORS
MOV      #12,$TIMES      ;DO 12 ITERATIONS
JSR      PC,CLRMAP      ;CLEAR ALL MAP REGISTERS
MOV      #TIMOUT,CPUEXP      ;TIMEOUTS MIGHT OCCUR IN THIS TEST.
MOV      #10$,$LPERR      ;SET LOOP ON ERROR POINTER TO 10$
MOV      #170400,KIPAR6      ;START WITH ADDRESS 8K FROM UNIBUS
CLR      PCPUER      ;CLEAR ERROR CONDITION LOCATION
NOP              ;THIS IS A SYNC POINT FOR SCOPING
MOV      @#140000,R0      ;TRY TO READ ADDRESS POINTED TO BY PAR6
TST      PCPUER      ;SEE IF READ OF ADDRESS TIMED OUT
BEQ      2$      ;BRANCH IF REFERENCE WAS GOOD
ADD      #200,KIPAR6      ;TRY NEXT 4K BLOCK OF MEMORY
CMP      #177600,KIPAR6      ;SEE IF YOU'VE POINTED TO I/O PAGE
BNE      1$      ;BRANCH IF NOT
ERROR    30      ;NO UNIBUS ADDRESSES RESPOND
JMP      SIZEJ      ;JUMP TO SIZE JUMPER TEST
MOV      KIPAR6,KIPAR5      ;PUT PAR6 INTO PAR5
BIC      #170000,KIPAR5      ;MAKE PAR5 A NON UNIBUS ADDRESS
MOV      #173214,@#120000      ;PUT RANDOM NUMBER INTO TEST LOCATION BY FAST BUS
MOV      @#140000,R1      ;READ TEST LOCATION VIA UNIBUS
CMP      #173214,R1      ;SEE IF DATA WAS READ PROPERLY
```



```
3052 014260 001403      BEQ      3$      ;DATA OKAY NOW VERIFY DATA PATH
3053 014262 020001      CMP      R0,R1    ;SEE IF DATA CHANGED FROM FIRST READ
3054 014264 001001      BNE      3$      ;BRANCH IF DATA CHANGED
3055 014266 000745      BR       4$      ;TRY NEXT 4K BLOCK
3056 014270 005001      CLR      R1      ;CLEAR REGISTER TO HOLD COUNT
3057 014272 012737 014304 001110 3$:    MOV      #11$,$LPERR ;SET LOOP ON ERROR POINTER TO 11$
3058 014300 012702 140000      MOV      #140000,R2 ;LOAD VIRTUAL ADDRESS INTO R2
3059 014304 000240      11$:    NOP      ;THIS IS A SYNC POINT FOR SCOPING
3060 014306 010112      5$:    MOV      R1,(R2) ;LOAD COUNT INTO TEST LOCATION VIA U.B.
3061 014310 011200      MOV      (R2),R0 ;READ TEST LOCATION BACK VIA UNIBUS
3062 014312 020100      CMP      R1,R0    ;COMPARE COUNT WITH DATA READ
3063 014314 001402      BEQ      6$      ;BRANCH IF DATA MATCHES
3064 014316 004737 005522      JSR      PC,UBCOUNT ;COUNT FAILED REPORT ERROR
3065 014322 005201      6$:    INC      R1    ;INCREASE COUNT
3066 014324 001370      BNE      5$      ;BRANCH IF COUNT HASN'T CYCLED
3067 014326 005737 001254      TST      ERRCNT ;WERE THERE ANY ERRORS ON THIS TEST
3068 014332 001401      BEQ      19$     ;BRANCH IF NO ERRORS ON THIS TEST
3069 014334 104031      ERROR    31      ;SUMMARY OF ERRORS ON THE UNIBUS
3070                                ;DATA PATH
3071 014336 012737 014134 001110 19$:  MOV      #20$,$LPERR ;SET LOOP POINTER TO START OF TEST
3072 014344 005037 001264      CLR      CPUEXP ;ZERO EXPECTED CPU TRAP CONDITION
3073
3074
3075
3076
3077
3078
3079
3080
3081
3082
3083
3084
3085
3086
3087
3088
```

```
*****
*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.
*****
```

```
3089 014350
3090 014350 000004      TST23:
3091 014352 012737 014462 001324      SCOPE
3092                                MOV      #TST24,NXITST ;SAVE STARTING ADDRESS OF NEXT TEST
3093 014360 012737 000020 001264 20$:  MOV      #TIMOUT,CPUEXP ;FOR ESCAPE ON PARITY ERRORS
3094 014366 012737 014424 001110      MOV      #10$,$LPERR ;MIGHT TIME OUT OVER UNIBUS
3095 014374 013700 172354      MOV      KIPAR6,R0 ;SET LOOP ON ERROR POINTER TO 10$
3096 014400 072027 177773      ASH      #-5,R0 ;PUT UNIBUS ADDRESS OF MAP REG IN R0
3097 014404 042700 007400      BIC      #007400,R0 ;RIGHT SHIFT R0 5 PLACES
3098 014410 012720 021042      MOV      #021042,(R0)+ ;STRIP OFF EXTRANEIOUS BITS.
3099 014414 012710 000042      MOV      #42,(R0) ;SET BOTTOM BIT IN EACH ALU
3100 014420 005037 120000      CLR      @#120000 ;SET BOTTOM BIT IN EACH ALU
3101 014424 000240      10$:    NOP      ;CLEAR TEST LOCATION VIA FAST BUS
3102 014426 012737 043207 140000      MOV      #43207,@#140000 ;THIS IS A SYNC POINT FOR SCOPING
3103                                ;LOAD TEST LOCATION VIA UNIBUS
3104                                ;THIS LOAD SHOULD NOT BE RELOCATED
3105                                ;BY THE UNIBUS MAP, SINCE BIT05 OF
3106                                ;MMR3 IS CLEAR.
3106 014434 013703 120000      MOV      @#120000,R3 ;READ TEST LOCATION VIA FAST BUS
3107 014440 022703 043207      CMP      #43207,R3 ;SEE IF DATA MATCHES
```

```
3108 014444 001401      BEQ      1$          ;BRANCH IF DATA GOOD
3109 014446 104032      ERROR    32          ;MAP RELOCATED WHEN NOT ENABLED
3110 014450 012737 014360 001110 1$:      MOV      #20$,$LPERR      ;SET LOOP POINTER TO START OF TEST
3111 014456 005037 001264      CLR      %CPUEXP      ;ZERO EXPECTED CPU TRAP CONDITION
3112
3113
3114
3115
3116
3117
3118
3119
3120
3121
3122
3123
3124
3125
3126
3127 014462
3128 014462 000004
3129 014464 012737 015120 001324
3130
3131 014472 012737 000001 001204
3132 014500
3133 014500 000004
3134 014502 012737 014532 001106
3135 014510 012737 014532 001110
3136 014516 012737 000024 001102
3137 014524 013737 001102 177570
3138 014532 012737 000020 001264 20$:
3139 014540 012700 170200
3140 014544 012720 020000 1$:
3141 014550 005020
3142 014552 022700 170374
3143 014556 101372
3144 014560 052737 000040 172516
3145 014566 012700 117776
3146 014572 012737 170000 172350
3147 014600 012701 000200
3148 014604 012702 125252
3149 014610 005037 037776 2$:
3150 014614 010210
3151 014616 023702 037776
3152 014622 001411
3153 014624 060137 172350
3154 014630 022737 177600 172350
3155 014636 001364
3156 014640 104033
3157 014642 000137 010000
3158 014646 013737 172350 001240 3$:
3159 014654 013737 172350 001242 4$:
3160 014662 060137 172350
3161 014666 022737 177600 172350
3162 014674 001433
3163 014676 005037 037776
```

```
*****
*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
      MOV      #1,$TIMES          ;DO 1 ITERATION
      SIZEJ:
      SCOPE
      MOV      #20$,$LPADR        ;SET LOOP ON TEST POINTER TO 20$
      MOV      #20$,$LPERR        ;SET LOOP ON ERROR POINTER TO 20$
      MOV      #24,$TSTNM         ;SETUP TEST NUMBER AND CLR ERROR FLAG
      MOV      $TSTNM,$DISPLAY    ;DISPLAY TEST NUMBER FOR ALL TO SEE
      MOV      #TIMOUT,$CPUEXP    ;EXPECTING CPU TIME OUT ON UNIBUS
      MOV      #MAPLO,$R0         ;LOAD ADDRESS OF FIRST MAP REG
      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      #BIT5,$MMR3        ;TURN ON MAP RELOCATION
      MOV      #117776,$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
      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
      MOV      $KIPAR4,$LOWEST    ;FOUND THE LOWEST USABLE MAP REG
      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
```


3164	014702	010210		MOV	R2,(R0)	;TRY TO LOAD TEST CELL THROUGH THE MAP
3165	014704	023702	037776	CMP	@#37776,R2	;SEE IF TEST LOCATION WAS LOADED
3166	014710	001761		BEQ	4\$	;BRANCH IF IT WAS LOADED
3167	014712	005037	037776	5\$: CLR	@#37776	;CLEAR TEST LOCATION
3168	014716	005237	001254	INC	ERRCNT	;COUNT OF REGS AFTER LAST ONE USABLE
3169	014722	010210		MOV	R2,(R0)	;TRY TO LOAD TEST CELL THRU MAP
3170	014724	023702	037776	CMP	@#37776,R2	;SEE IF TEST LOCATION WAS LOADED
3171	014730	001402		BEQ	6\$	;BRANCH IF LOCATION WAS LOADED

PDP-11/70-74MP UNIBUS MAP DIAGNOSTIC
CEKBFC.P11 03-MAY-79 13:19

MACY11 30A(1052) 03-MAY-79 13:21 PAGE 63

T24 SIZE JUMPER LOCATION

SEQ 0074

3172	014732	005237	001256		INC	CNTR	:COUNT OF REGS THAT FAILED TO RESPOND
3173	014736	060137	172350		ADD	R1,KIPAR4	:POINT TO NEXT MAP REG UNDER TEST
3174	014742	022737	177600	172350	CMP	#177600,KIPAR4	:SEE IF TEST IS OVER
3175	014750	001360			BNE	5\$	:BRANCH IF TEST NOT DONE
3176	014752	023737	001254	001256	CMP	ERRCNT,CNTR	:SEE IF TRIES = FAILURES
3177	014760	001401			BEQ	7\$	:BRANCH IF EQUAL
3178	014762	104034			ERROR	34	:MAP JUMPER COMPARE CIRCUIT BAD

3179	014764	005037	001254	7\$:	CLR	ERRCNT	:CLEAR TRIES COUNTER
3180	014770	005037	001256		CLR	CNTR	:CLEAR FAILURE COUNTER
3181	014774	005037	001264		CLR	CPUEXP	:NO CPU TRAPS EXPECTED IN NEXT TEST
3182	015000	005737	001100		TST	\$PASS	:SEE IF THIS IS FIRST PASS
3183	015004	001045			BNE	TST25	:GO TO NEXT TEST IF NOT THE FIRST PASS
3184	015006	023727	001240 170000		CMP	LOWEST,#170000	:SEE IF LOWER JUMPER IS DEFAULT
3185	015014	001004			BNE	8\$	:BRANCH IF NOT.
3186	015016	023727	001242 177400		CMP	HIGEST,#177400	:SEE IF UPPER JUMPER IS DEFAULT.
3187	015024	001401			BEQ	9\$	:BRANCH IF JUMPERS DEFAULT.
3188	015026	104035		8\$:	ERROR	35	:MAP SIZE JUMPERS NOT DEFAULT
3189				::			
3190				::*			
3191				::*			
3192				:::			
3193	015030	013700	001242	9\$:	MOV	HIGEST,R0	
3194	015034	013701	001240		MOV	LOWEST,R1	
3195	015040	042700	170000		BIC	#170000,R0	
3196	015044	042701	170000		BIC	#170000,R1	
3197	015050	072027	177773		ASH	#-5,R0	:RIGHT SHIFT R0 5 PLACES
3198	015054	072127	177773		ASH	#-5,R1	:RIGHT SHIFT R1 5 PLACES
3199	015060	062701	170200		ADD	#170200,R1	
3200	015064	062700	170200		ADD	#170200,R0	
3201	015070	010137	001244		MOV	R1,LREGL	
3202	015074	062701	000002		ADD	#2,R1	:POINT TO UPPER BITS OF MAP REG
3203	015100	010137	001246		MOV	R1,LREGU	
3204	015104	010037	001250		MOV	R0,HREGL	
3205	015110	062700	000002		ADD	#2,R0	:POINT TO UPPER BITS OF MAP REG
3206	015114	010037	001252		MOV	R0,HREGU	
3207							
3208							
3209							

3210
3211
3212
3213
3214
3215
3216
3217
3218
3219
3220
3221
3222
3223
3224
3225
3226
3227
3228
3229
3230
3231
3232
3233
3234
3235
3236
3237
3238
3239
3240
3241
3242
3243
3244
3245
3246
3247
3248
3249
3250
3251
3252
3253
3254
3255
3256
3257
3258
3259
3260
3261
3262
3263
3264
3265

```

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

TST25:
SCOPE
MOV      #TST26,NXTTST      ;SAVE STARTING ADDRESS OF NEXT TEST
                                ;FOR ESCAPE ON PARITY ERRORS
                                ;DO 144 ITERATIONS
20$:     MOV      #144,$TIMES
        JSR      PC,CLRMAP    ;CLEAR ALL MAP REGISTERS
        CLR      $TMP0        ;USED AS FLAG IN TEST
        MOV      #100000,R3    ;SELECT P.A.R. 4 OFFSET OF ZERO
        MOV      #2$,SLPERR    ;SET LOOP ON ERROR POINTER TO 2$
        MOV      LREGL,R2      ;PUT ADDRESS OF LOWEST USABLE MAP REG IN R2
        MOV      LOWEST,R0     ;MAP REGISTER UNDER TEST IN R0
        MOV      LOWEST,R1     ;MAP REGISTER USED IN CURRENT REFERENCE
1$:      MOV      #40000,(R2)   ;LOAD MAP REG UNDER TEST WITH 8K BASE
        MOV      R2,#40000     ;LOAD TEST LOCATION WITH THE ADDRESS
                                ;OF THE MAP REGISTER UNDER TEST
        MOV      R1,KIPAR4     ;LOAD PAR 4 WITH NEXT MAP REG UNIBUS ADDR
        MOV      (R3),R4       ;READ THROUGH THE MAP
        CMP      R4,R2         ;SEE IF CORRECT DATA WAS FETCHED
        BNE      4$            ;BRANCH IF NO MATCH
        CMP      R0,R1         ;SEE IF MAP REGS ARE THE SAME
        BEQ      3$            ;BRANCH IF CORRECT MAP REG WAS USED
        JSR      PC,DUALADR    ;LOG AND REPORT ALL ERRORS
        BR       4$           ;SKIP NEXT INSTRUCTION
        MOV      #1,$TMP0      ;SET FLAG WHEN ADDRS MATCH
4$:      ADD      #200,R1       ;TRY NEXT MAP REG
        CMP      HIGEST,R1     ;SEE IF ALL HAVE BEEN TRIED
        BHIS     2$            ;BRANCH IF STILL MORE TO TRY
        TST      $TMP0         ;SEE THAT THERE WAS A SUCCESSFUL MATCH
        BNE      5$           ;BRANCH IF THERE WAS
        ERROR    36            ;DID NOT MATCH MAP REG ADDRESS
5$:      CLR      $TMP0        ;CLEAR FLAG FOR NEXT REG
        CLR      (R2)          ;CLEAR MAP REG JUST TESTED
        ADD      #4,R2         ;POINT TO NEXT MAP REG TO LOAD
        ADD      #200,R0       ;POINT TO NEXT MAP REG UNDER TEST
        CMP      HIGEST,R0     ;SEE IF ALL MAP REGS HAVE BEEN TESTED
        BHIS     1$           ;BRANCH IF STILL MORE TO TEST
        TST      ERRCNT        ;SEE IF THERE WERE ANY ERRORS
        BEQ      TST26        ;BRANCH TO NEXT TEST IF NO ERRORS
        ERROR    13            ;ERROR TYPE OUT ITEM 13

```


3266
3267
3268
3269
3270
3271
3272
3273
3274
3275
32763277 015314
3278 015314 000004
3279 015316 012737 015574 001324
3280
3281 015324 042737 000040 172516
3282 015332 012737 000400 172350
3283 015340 012700 040000
3284 015344 012701 100000
3285 015350 012702 010000
3286 015354 010021
3287 015356 062700 000002
3288 015362 077204
3289 015364 062737 000200 172350
3290 015372 022737 001400 172350
3291 015400 1013613292
3293
3294
3295 015402 012737 015410 001106
3296 015410 012737 015446 001110
3297 015416 022737 171400 172354
3298
3299 015424 101463
3300 015426 013700 172354
3301
3302 015432 072027 000006
3303 015436 012701 140000
3304 015442 012702 010000
3305 015446 000240
3306 015450 011103
3307 015452 020003
3308 015454 001015
3309 015456 062701 000002
3310 015462 062700 000002
3311 015466 077211
3312 015470 062737 000200 172354
3313 015476 022737 171400 172354
3314 015504 101354
3315 015506 0004233316
3317 015510
3318 015510 005100
3319 015512 040037 001224
3320 015516 005100
3321 015520 005103*****
: *TEST 26 LOAD LOCATIONS 40000 - 137776 WITH THEIR ADDRESSES
: *
: * THIS TEST IS USED TO LOAD MAIN MEMORY FROM ADDRESS 00040000 TO
: * ADDRESS 00137776 WITH ITS OWN ADDRESS. IT THEN CHECKS THAT
: * MEMORY OVER THE UNIBUS AND LOGS AND REPORTS ANY ERRORS THAT
: * IT FINDS.
: *
: *****

TST26:

SCOPE
MOV #TST27,NXTTST ;SAVE STARTING ADDRESS OF NEXT TEST
;FOR ESCAPE ON PARITY ERRORS
BIC #BIT5,MMR3 ;TURN OFF MAP RELOCATION
MOV #400,KIPAR4 ;MAP PAGE 4 TO 8K
MOV #40000,R0 ;STARTING ADDRESS FOR DATA PATTERN
1\$: MOV #100000,R1 ;VIRTUAL ADDRESS
MOV #D4096,R2 ;LOAD 4096 LOCATIONS AT A TIME
2\$: MOV R0,(R1)+ ;LOAD PHY. ADDR. INTO EACH MEMORY LOC.
ADD #2,R0 ;POINT TO NEXT PHYSICAL ADDRESS
SOB R2,2\$;BRANCH IF 4K OF MEMORY NOT LOADED
ADD #200,KIPAR4 ;POINT TO NEXT 4K BANK OF MEMORY
CMP #1400,KIPAR4 ;SEE IF 24K IS LOADED
BHI 1\$;BRANCH IF MORE MEMORY TO LOAD: *
: * MEMORY FROM 8K - 24K IS NOW LOADED WITH ITS OWN ADDRESS
: *20\$: MOV #20\$,\$LPADR ;SET LOOP ADDRESS TO 20\$
MOV #4\$,\$LPERR ;SET LOOP ON ERROR POINTER TO 4\$
CMP #171400,KIPAR6 ;DID I USE ANY MAP REGISTER
;BELOW REGISTER 6 (UB. ADDR 140000)
BLOS TST27 ;BRANCH TO NEXT TEST IF NOT
MOV KIPAR6,R0 ;LOAD PAR6 INTO R0 TO GET
;THE STARTING DATA PATTERN
3\$: ASH #6,R0 ;R0 NOW HOLDS THE STARTING DATA PATTERN
MOV #140000,R1 ;STARTING VIRTUAL ADDRESS
MOV #D4096,R2 ;PREPARE TO READ 4K AT A TIME
4\$: NOP ;THIS IS A SYNC POINT FOR SCOPING
MOV (R1),R3 ;READ MAIN MEMORY THRU UNIBUS
CMP R0,R3 ;SEE IF THE ADDRESSES MATCH
BNE 6\$;BRANCH IF ERROR
5\$: ADD #2,R1 ;CHANGE VIRTUAL ADDRESS
ADD #2,R0 ;CHANGE PHYSICAL ADDRESS
SOB R2,4\$;BRANCH IF 4K OF MEMORY NOT READ
ADD #200,KIPAR6 ;POINT TO NEXT BANK OF 4K THRU UNIBUS
CMP #171400,KIPAR6 ;SEE IF THIS POINTS TO 24K PLUS 2
BHI 3\$;BRANCH IF 24K OF MEMORY NOT CHECKED
BR 10\$;TEST FINISHED, BRANCH TO EXIT6\$: COM R0 ;GET R0 READY FOR AND
BIC R0,ADRAND ;PERFORM LOGICAL AND
COM R0 ;PUT R0 BACK AS IT WAS
COM R3 ;GET R3 READY FOR AND

3322	015522	040337	001230	BIC	R3,DATAND	:PERFORM LOGICAL AND	
3323	015526	005103		COM	R3	:PUT R3 BACK AS IT WAS	
3324	015530	050037	001226	BIS	R0,ADDROR	:LOGICAL OR OF PHYSICAL ADDRESS	
3325	015534	050337	001232	BIS	R3,DATAOR	:LOGICAL OR OF ADDRESS FETCHED	
3326	015540	005737	001254	TST	ERRCNT	:IS THIS THE FIRST ERROR HERE?	
3327	015544	001002		BNE	7\$	:BRANCH IF NOT FIRST ERROR	
3328	015546	104206		ERROR	206	:DIDN'T READ ADDRESSES RIGHT FROM UNIBUS	
3329	015550	000742		BR	5\$	:CONTINUE TESTING	
3330	015552	104306	7\$:	ERROR	306	:REPORT MORE DATA FROM UNIBUS	
3331	015554	000740		BR	5\$	:CONTINUE WITH TEST	
3332	015556	005737	001254	10\$:	TST	ERRCNT	:WERE THERE ANY ERRORS ON THIS TEST?
3333	015562	001401		BEQ	19\$	:BRANCH IF NO ERRORS ON THIS TEST	
3334	015564	104037		ERROR	37	:SUMMARY OF UNIBUS ADDRESS FAILURES	
3335	015566	012737	015410 001110	19\$:	MOV	#20\$,\$LPERR	:SET LOOP POINTER TO 20\$

```

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

```

ADDRESS	HEX	HEX	HEX	HEX	HEX	INSTR	COMMENT
3347	015574					IST27:	
3348	015574	000004				SCOPE	
3349	015576	012737	015706	001324		MOV	#TST30,NXTTST ;SAVE STARTING ADDRESS OF NEXT TEST
3350							;FOR ESCAPE ON PARITY ERRORS
3351	015604	052737	000040	172516	20\$:	BIS	#BIT5,MMR3 ;TURN MAP RELOCATION BACK ON
3352	015612	012737	000020	001264		MOV	#TIMOUT,CPUEXP ;EXPECTING TIMEOUT IN THIS TEST
3353	015620	013737	001240	172350		MOV	LOWEST,KIPAR4 ;LOAD PAR 4 WITH LOWEST USABLE MAP REG
3354	015626	012777	000074	163412		MOV	#74,@LREGU ;LOAD UPPER 6 BITS OF LOWEST MAP REG
3355	015634	012777	000000	163402		MOV	#000000,@LREGL ;LOAD LOWER 16 BITS OF LOWEST MAP REG
3356	015642	012737	015650	001110		MOV	#1\$,\$LPERR ;SET LOOP ON ERROR POINTER TO 1\$
3357	015650	005037	001266		1\$:	CLR	PCPUER ;CPU ERROR REG LOCATION
3358	015654	000240				NOP	;THIS IS A SYNC POINT FOR SCOPING
3359	015656	013703	100000			MOV	@#100000,R3 ;TRY TO READ THRU PAGE 4
3360							;THIS REFERENCE WILL GO OUT ON THE
3361							UNIBUS TO SELECT THE LOWEST USEABLE
3362							MAP REGISTER (DEFAULT MAP REG. 0).
3363							PHYSICAL ADDRESS 17700000 IS THEN
3364							GENERATED, WHICH SHOULD TIME OUT SINCE
3365							IT IS THE FIRST NON-EXISTANT LOCATION.
3366	015662	022737	000020	001266		CMP	#TIMOUT,PCPUER ;THE UNIBUS SHOULD HAVE TIMED OUT
3367	015670	001401				BEQ	10\$;BRANCH IF CONDITION WAS CORRECT
3368	015672	104040				ERROR	40 ;UNIBUS DID NOT TIME OUT
3369	015674	012737	015604	001110	10\$:	MOV	#20\$,\$LPERR ;SET LOOP POINTER TO START OF TEST
3370	015702	005037	001264			CLR	CPUEXP ;NO CPU TRAPS EXPECTED FOR AWHILE

```

:*****
:*TEST 30          RELOCATION TEST USING LOWEST USABLE MAPPING REG
:*
:*
:*      THIS TEST CHECKS OUT THE FULL ADDITION PROPERTIES OF THE UNIBUS
:*      MAP A.L.U..  IN THE DEFAULT CASE IT USES MAP REGISTER ZERO BUT

```



```
3378      IF THE MAP JUMPERS HAVE BEEN ALTERED TO DE-SELECT SOME MAP REGISTERS
3379      THIS TEST WILL USE THE LOWEST USEABLE MAP REGISTER.
3380      IF AN ERROR OCCURS THE TEST WILL REPORT THE PHYSICAL ADDRESS
3381      THAT WAS DESIRED, AND THE DATA AT THE ADDRESS THAT WAS REFERENCED.
3382
3383      *****
3384      TST30:
3385      SCOPE
3386      MOV      #TST31,NXTTST      ;SAVE STARTING ADDRESS OF NEXT TEST
3387      ;FOR ESCAPE ON PARITY ERRORS
3388
3389      ;:
3390      ;*THE RELOCATION HERE USES A BASE OF 00060000 AND AN
3391      ;*OFFSET OF 00000 TO PRODUCE AN ADDRESS OF 00060000
3392      100$:  MOV      #1$, $LPERR      ;SET LOOP ON ERROR POINTER TO 1$
3393      CLR      @LREGU      ;CLEAR UPPER BITS OF MAPPING REG
3394      MOV      #060000, @LREGL      ;LOAD LOWER BITS OF MAPPING REG
3395      MOV      LOWEST, @#KIPAR6      ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
3396      MOV      #140000, R0      ;SELECT PAR6, OFFSET IS 00000
3397      1$:    NOP
3398      MOV      (R0), R1      ;THIS IS A SYNC POINT FOR SCOPING
3399      MOV      #060000, R2      ;READ LOCATION 060000 THRU THE UNIBUS
3400      CMP      R1, R2      ;THE EXPECTED PHYSICAL ADDRESS IS 060000
3401      BEQ      2$      ;SEE IF THE MAP'S FETCH WAS CORRECT
3402      ERROR    41      ;BRANCH IF FETCHED DATA MATCHES ADDRESS
3403      ;FAULTY RELOCATION BY UNIBUS MAP
3404
3405      ;:
3406      ;*THE RELOCATION HERE USES A BASE OF 00052524 AND AN
3407      ;*OFFSET OF 05252 TO PRODUCE AN ADDRESS OF 00057776
3408      2$:    MOV      #3$, $LPERR      ;SET LOOP ON ERROR POINTER TO 3$
3409      CLR      @LREGU      ;CLEAR UPPER BITS OF MAPPING REG
3410      MOV      #052524, @LREGL      ;LOAD LOWER BITS OF MAPPING REG
3411      MOV      LOWEST, @#KIPAR6      ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
3412      MOV      #145252, R0      ;SELECT PAR6, OFFSET IS 05252
3413      3$:    NOP
3414      MOV      (R0), R1      ;THIS IS A SYNC POINT FOR SCOPING
3415      MOV      #057776, R2      ;READ LOCATION 057776 THRU THE UNIBUS
3416      CMP      R1, R2      ;THE EXPECTED PHYSICAL ADDRESS IS 057776
3417      BEQ      4$      ;SEE IF THE MAP'S FETCH WAS CORRECT
3418      ERROR    41      ;BRANCH IF FETCHED DATA MATCHES ADDRESS
3419      ;FAULTY RELOCATION BY UNIBUS MAP
3420
3421      ;:
3422      ;*THE RELOCATION HERE USES A BASE OF 00045252 AND AN
3423      ;*OFFSET OF 12524 TO PRODUCE AN ADDRESS OF 00057776
3424      4$:    MOV      #5$, $LPERR      ;SET LOOP ON ERROR POINTER TO 5$
3425      CLR      @LREGU      ;CLEAR UPPER BITS OF MAPPING REG
3426      MOV      #045252, @LREGL      ;LOAD LOWER BITS OF MAPPING REG
3427      MOV      LOWEST, @#KIPAR6      ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
3428      MOV      #152524, R0      ;SELECT PAR6, OFFSET IS 12524
3429      5$:    NOP
3430      MOV      (R0), R1      ;THIS IS A SYNC POINT FOR SCOPING
3431      MOV      #057776, R2      ;READ LOCATION 057776 THRU THE UNIBUS
3432      CMP      R1, R2      ;THE EXPECTED PHYSICAL ADDRESS IS 057776
3433      BEQ      6$      ;SEE IF THE MAP'S FETCH WAS CORRECT
3434      ERROR    41      ;BRANCH IF FETCHED DATA MATCHES ADDRESS
3435      ;FAULTY RELOCATION BY UNIBUS MAP
```

```
3434
3435
3436
3437 016106 012737 016140 001110 6$: MOV #7$, $LPERR ;SET LOOP ON ERROR POINTER TO 7$
3438 016114 005077 163126 CLR @LREGU ;CLEAR UPPER BITS OF MAPPING REG
3439 016120 012777 050420 163116 MOV #050420, @LREGL ;LOAD LOWER BITS OF MAPPING REG
3440 016126 013737 001240 172354 MOV LOWEST, @#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
3441 016134 012700 150420 MOV #150420, R0 ;SELECT PAR6, OFFSET IS 10420
3442 016140 000240 7$: NOP ;THIS IS A SYNC POINT FOR SCOPING
3443 016142 011001 MOV (R0), R1 ;READ LOCATION 061040 THRU THE UNIBUS
3444 016144 012702 061040 MOV #061040, R2 ;THE EXPECTED PHYSICAL ADDRESS IS 061040
3445 016150 020102 CMP R1, R2 ;SEE IF THE MAP'S FETCH WAS CORRECT
3446 016152 001401 BEQ 8$ ;BRANCH IF FETCHED DATA MATCHES ADDRESS
3447 016154 104041 ERROR 41 ;FAULTY RELOCATION BY UNIBUS MAP
3448
3449
3450
3451
3452 016156 012737 016210 001110 8$: MOV #9$, $LPERR ;SET LOOP ON ERROR POINTER TO 9$
3453 016164 005077 163056 CLR @LREGU ;CLEAR UPPER BITS OF MAPPING REG
3454 016170 012777 054630 163046 MOV #054630, @LREGL ;LOAD LOWER BITS OF MAPPING REG
3455 016176 013737 001240 172354 MOV LOWEST, @#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
3456 016204 012700 144210 MOV #144210, R0 ;SELECT PAR6, OFFSET IS 04210
3457 016210 000240 9$: NOP ;THIS IS A SYNC POINT FOR SCOPING
3458 016212 011001 MOV (R0), R1 ;READ LOCATION 061040 THRU THE UNIBUS
3459 016214 012702 061040 MOV #061040, R2 ;THE EXPECTED PHYSICAL ADDRESS IS 061040
3460 016220 020102 CMP R1, R2 ;SEE IF THE MAP'S FETCH WAS CORRECT
3461 016222 001401 BEQ 10$ ;BRANCH IF FETCHED DATA MATCHES ADDRESS
3462 016224 104041 ERROR 41 ;FAULTY RELOCATION BY UNIBUS MAP
3463
3464
3465
3466
3467 016226 012737 016260 001110 10$: MOV #11$, $LPERR ;SET LOOP ON ERROR POINTER TO 11$
3468 016234 005077 163006 CLR @LREGU ;CLEAR UPPER BITS OF MAPPING REG
3469 016240 012777 044210 162776 MOV #044210, @LREGL ;LOAD LOWER BITS OF MAPPING REG
3470 016246 013737 001240 172354 MOV LOWEST, @#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
3471 016254 012700 154630 MOV #154630, R0 ;SELECT PAR6, OFFSET IS 14630
3472 016260 000240 11$: NOP ;THIS IS A SYNC POINT FOR SCOPING
3473 016262 011001 MOV (R0), R1 ;READ LOCATION 061040 THRU THE UNIBUS
3474 016264 012702 061040 MOV #061040, R2 ;THE EXPECTED PHYSICAL ADDRESS IS 061040
3475 016270 020102 CMP R1, R2 ;SEE IF THE MAP'S FETCH WAS CORRECT
3476 016272 001401 BEQ 12$ ;BRANCH IF FETCHED DATA MATCHES ADDRESS
3477 016274 104041 ERROR 41 ;FAULTY RELOCATION BY UNIBUS MAP
3478
3479
3480
3481
3482 016276 012737 016330 001110 12$: MOV #13$, $LPERR ;SET LOOP ON ERROR POINTER TO 13$
3483 016304 005077 162736 CLR @LREGU ;CLEAR UPPER BITS OF MAPPING REG
3484 016310 012777 056734 162726 MOV #056734, @LREGL ;LOAD LOWER BITS OF MAPPING REG
3485 016316 013737 001240 172354 MOV LOWEST, @#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
3486 016324 012700 142104 MOV #142104, R0 ;SELECT PAR6, OFFSET IS 02104
3487 016330 000240 13$: NOP ;THIS IS A SYNC POINT FOR SCOPING
3488 016332 011001 MOV (R0), R1 ;READ LOCATION 061040 THRU THE UNIBUS
3489 016334 012702 061040 MOV #061040, R2 ;THE EXPECTED PHYSICAL ADDRESS IS 061040
```



```
3490 016340 020102      CMP      R1,R2      ;SEE IF THE MAP'S FETCH WAS CORRECT
3491 016342 001401      BEQ      14$      ;BRANCH IF FETCHED DATA MATCHES ADDRESS
3492 016344 104041      ERROR    41      ;FAULTY RELOCATION BY UNIBUS MAP
3493
3494      ;;
3495      ;*THE RELOCATION HERE USES A BASE OF 00042104 AND AN
3496      ;*OFFSET OF 16734 TO PRODUCE AN ADDRESS OF 00061040
3497 016346 012737 016400 001110 14$:  MOV      #15$,$LPERR      ;SET LOOP ON ERROR POINTER TO 15$
3498 016354 005077 162666      CLR      @LREGU      ;CLEAR UPPER BITS OF MAPPING REG
3499 016360 012777 042104 162656      MOV      #042104,@LREGL      ;LOAD LOWER BITS OF MAPPING REG
3500 016366 013737 001240 172354      MOV      LOWEST,@#KIPAR6      ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
3501 016374 012700 156734      MOV      #156734,R0      ;SELECT PAR6, OFFSET IS 16734
3502 016400 000240      15$:  NOP      ;THIS IS A SYNC POINT FOR SCOPING
3503 016402 011001      MOV      (R0),R1      ;READ LOCATION 061040 THRU THE UNIBUS
3504 016404 012702 061040      MOV      #061040,R2      ;THE EXPECTED PHYSICAL ADDRESS IS 061040
3505 016410 020102      CMP      R1,R2      ;SEE IF THE MAP'S FETCH WAS CORRECT
3506 016412 001401      BEQ      16$      ;BRANCH IF FETCHED DATA MATCHES ADDRESS
3507 016414 104041      ERROR    41      ;FAULTY RELOCATION BY UNIBUS MAP
3508
3509      ;;
3510      ;*THE RELOCATION HERE USES A BASE OF 00057776 AND AN
3511      ;*OFFSET OF 01042 TO PRODUCE AN ADDRESS OF 00061040
3512 016416 012737 016450 001110 16$:  MOV      #17$,$LPERR      ;SET LOOP ON ERROR POINTER TO 17$
3513 016424 005077 162616      CLR      @LREGU      ;CLEAR UPPER BITS OF MAPPING REG
3514 016430 012777 057776 162606      MOV      #057776,@LREGL      ;LOAD LOWER BITS OF MAPPING REG
3515 016436 013737 001240 172354      MOV      LOWEST,@#KIPAR6      ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
3516 016444 012700 141042      MOV      #141042,R0      ;SELECT PAR6, OFFSET IS 01042
3517 016450 000240      17$:  NOP      ;THIS IS A SYNC POINT FOR SCOPING
3518 016452 011001      MOV      (R0),R1      ;READ LOCATION 061040 THRU THE UNIBUS
3519 016454 012702 061040      MOV      #061040,R2      ;THE EXPECTED PHYSICAL ADDRESS IS 061040
3520 016460 020102      CMP      R1,R2      ;SEE IF THE MAP'S FETCH WAS CORRECT
3521 016462 001401      BEQ      18$      ;BRANCH IF FETCHED DATA MATCHES ADDRESS
3522 016464 104041      ERROR    41      ;FAULTY RELOCATION BY UNIBUS MAP
3523
3524      ;;
3525      ;*THE RELOCATION HERE USES A BASE OF 00041042 AND AN
3526      ;*OFFSET OF 17776 TO PRODUCE AN ADDRESS OF 00061040
3527 016466 012737 016520 001110 18$:  MOV      #19$,$LPERR      ;SET LOOP ON ERROR POINTER TO 19$
3528 016474 005077 162546      CLR      @LREGU      ;CLEAR UPPER BITS OF MAPPING REG
3529 016500 012777 041042 162536      MOV      #041042,@LREGL      ;LOAD LOWER BITS OF MAPPING REG
3530 016506 013737 001240 172354      MOV      LOWEST,@#KIPAR6      ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
3531 016514 012700 157776      MOV      #157776,R0      ;SELECT PAR6, OFFSET IS 17776
3532 016520 000240      19$:  NOP      ;THIS IS A SYNC POINT FOR SCOPING
3533 016522 011001      MOV      (R0),R1      ;READ LOCATION 061040 THRU THE UNIBUS
3534 016524 012702 061040      MOV      #061040,R2      ;THE EXPECTED PHYSICAL ADDRESS IS 061040
3535 016530 020102      CMP      R1,R2      ;SEE IF THE MAP'S FETCH WAS CORRECT
3536 016532 001401      BEQ      20$      ;BRANCH IF FETCHED DATA MATCHES ADDRESS
3537 016534 104041      ERROR    41      ;FAULTY RELOCATION BY UNIBUS MAP
3538
3539      ;;
3540      ;*THE RELOCATION HERE USES A BASE OF 00057776 AND AN
3541      ;*OFFSET OF 00002 TO PRODUCE AN ADDRESS OF 00060000
3542 016536 012737 016570 001110 20$:  MOV      #21$,$LPERR      ;SET LOOP ON ERROR POINTER TO 21$
3543 016544 005077 162476      CLR      @LREGU      ;CLEAR UPPER BITS OF MAPPING REG
3544 016550 012777 057776 162466      MOV      #057776,@LREGL      ;LOAD LOWER BITS OF MAPPING REG
3545 016556 013737 001240 172354      MOV      LOWEST,@#KIPAR6      ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
```

```
3546 016564 012700 140002      MOV      #140002,R0      :SELECT PAR6, OFFSET IS 00002
3547 016570 000240      21$:  NOP              :THIS IS A SYNC POINT FOR SCOPING
3548 016572 011001      MOV      (R0),R1      :READ LOCATION 060000 THRU THE UNIBUS
3549 016574 012702 060000  MOV      #060000,R2      :THE EXPECTED PHYSICAL ADDRESS IS 060000
3550 016600 020102      CMP      R1,R2      :SEE IF THE MAP'S FETCH WAS CORRECT
3551 016602 001401      BEQ      22$      :BRANCH IF FETCHED DATA MATCHES ADDRESS
3552 016604 104041      ERROR    41      :FAULTY RELOCATION BY UNIBUS MAP
3553 016606 012737 015716 001110 22$:  MOV      #100$,$LPERR      :SET LOOP POINTER TO START OF TEST
3554
3555
3556      :*****
3557      :*TEST 31      TEST CARRY PROPAGATION OF MAP'S RELOCATION ADDER
3558      :*
3559      :*      EVERY ADDRESS OF THE FORM XXXX0000 IS GENERATED HERE STARTING
3560      :*      WITH 00030000 UP TO 17000000. THAT IS, THE FIRST OF EVERY 2K
3561      :*      WORDS IS ADDRESSED, TO INSURE THAT THE ADDER IN THE MAP IS
3562      :*      WORKING PROPERLY AND, THE SYSTEM SIZE JUMPERS ARE ACTUALLY SET
3563      :*      FOR THE TOP OF MAIN MEMORY.
3564      :*
3565      :*****
3566      TST31:
3567 016614 000004      SCOPE
3568 016616 012737 017120 001324  MOV      #TST32,NXTTST      :SAVE STARTING ADDRESS OF NEXT TEST
3569      :FOR ESCAPE ON PARITY ERRORS
3570 016624 012737 000012 001204  MOV      #12,$TIMES      :DO 12 ITERATIONS
3571      .EQUIV      BIT5,NEXMEM      :BIT05 IS NON-EXISTANT MEMORY BIT IN
3572      :THE CPU ERROR REGISTER
3573 016632 012737 177777 016766 20$:  MOV      #-1,4$      :INITIALIZE FLAG AS NEGATIVE ONE
3574 016640 005077 162402      CLR      @LREGU      :CLEAR UPPER 6 BITS OF MAP REG
3575 016644 012777 020000 162372  MOV      #20000,@LREGL      :LOAD 4K BASE INTO MAP REGISTER
3576 016652 012701 100100      MOV      #100100,R1      :LOAD BITS TO SELECT PAR 4, OFFSET 100
3577 016656 012700 150000      MOV      #150000,R0      :LOAD BITS TO SELECT PAR 6, OFFSET 2K
3578 016662 012737 000277 172350  MOV      #277,KIPAR4      :START WITH PHYSICAL 6K
3579 016670 013737 001240 172354  MOV      LOWEST,KIPAR6      :LOAD PAR 6 WITH MAP REG'S ADDR
3580 016676 012737 016736 001110  MOV      #10$,$LPERR      :SET LOOP ON ERROR POINTER TO 10$
3581 016704 012737 000020 001264 3$:  MOV      #20,CPUEXP      :EXPECTING A UNIBUS TIME OUT DURING TEST
3582 016712 005037 001266      CLR      PCPUER      :CLEAR TIME OUT FLAG
3583 016716 013710 001326      MOV      DATA,(R0)      :THIS LOAD WILL TIME OUT WHEN YOU
3584      :HAVE REACHED THE TOP OF MEMORY
3585      :IT SELECTS PAR 6 WHICH WILL PUT ADDR
3586      :<XXX XX1>0000 ON THE UNIBUS.
3587      :THE X'S WILL SELECT THE LOWEST USEABLE
3588      :MAPPING REGISTER. THE DEFAULT CASE IS
3589      :010000, SELECTING MAPPING REGISTER 0.
3590 016722 005737 001266      TST      PCPUER      :SEE IF THERE WAS MAIN MEMORY
3591 016726 001016      BNE      1$      :BRANCH IF NO MAIN MEMORY FROM UNIBUS
3592 016730 012737 000040 001264  MOV      #NEXMEM,CPUEXP      :POSSIBLE CACHE NON-EXISTANT MEMORY
3593 016736 011103      10$:  MOV      (R1),R3      :READ TEST LOCATION VIA FASTBUS
3594 016740 022737 000040 001266  CMP      #NEXMEM,PCPUER      :WAS THIS CACHE NON-EXISTANT MEMORY
3595 016746 001414      BEQ      2$      :BRANCH IF NON-EXSITANT MEMORY
3596 016750 000240      NOP              :THIS IS A SYNC POINT FOR SCOPING
3597 016752 011002      MOV      (R0),R2      :READ TEST LOCATION VIA UNIBUS MAP
3598 016754 020203      CMP      R2,R3      :COMPARE TEST DATA R2=MAP DATA
3599      :R3=FASTBUS DATA
3600 016756 001410      BEQ      2$      :BRANCH IF IT WAS THE SAME
3601 016760 104042      ERROR    42      :CARRY PROPAGATION FAILURE IN MAP
```


3602	016762	000406			BR	2\$	:BRANCH TO UPDATE ROUTINE
3603	016764	005227			1\$: INC	(PC)+	:INCREMENT ONE TIME ENTRANCE FLAG
3604	016766	177777			4\$: .WORD	-1	:USE NEGATIVE ONE FOR FLAG
3605	016770	001003			BNE	2\$	:BRANCH IF YOU'VE BEEN HERE BEFORE
3606	016772	013737	172350	001320	MOV	KIPAR4,RSIZE	:SAVE UPPER LIMIT OF MEMORY
3607	017000	062737	000100	001326	2\$: ADD	#100,DATA	:CHANGE PATTERN FOR NEXT LOAD
3608	017006	062737	000100	172350	ADD	#100,KIPAR4	:ADD 2K TO PAR4
3609	017014	062777	010000	162222	ADD	#10000,@LREGL	:ADD 2K TO MAP REGISTER
3610	017022	001330			BNE	3\$	:BRANCH IF MAP REGISTER NOT ZERO
3611	017024	005277	162216		INC	@LREGU	:ADD ONE TO UPPER 6 BITS OF MAP REG
3612	017030	022777	000073	162210	CMP	#73,@LREGU	:SEE IF TOP 128K BLOCK HAS BEEN PASSED
3613	017036	103322			BHIS	3\$	:BRANCH IF NOT PAST IT
3614	017040	005237	001326		INC	DATA	:CHANGE DATA PATTERN FOR NEXT PASS
3615	017044	042737	177700	001326	BIC	#177700,DATA	:CLEAR UPPER 10 BITS OF DATA PATTERN
3616	017052	052737	000300	001326	BIS	#300,DATA	:START WITH 3XX IN DATA PATTERN
3617	017060	012737	016632	001110	MOV	#20\$,\$LPERR	:SET LOOP POINTER TO START OF TEST
3618	017066	023737	177760	001320	CMP	@#SIZELO,RSIZE	:SEE IF SIZE JUMPERS AGREE WITH MEMORY TOP
3619	017074	001407			BEQ	19\$	:BRANCH IF TOP OF MEMORY AGREES WITH
3620							:THE SIZE JUMPERS
3621	017076	105737	001331		TSTB	KB11EM	:RUNNING ON A KB11-EM?
3622	017102	001004			BNE	19\$	:SKIP ERROR IF YES
3623	017104	105737	001332		TSTB	KB11CM	:RUNNING ON MODIFIED MACHINE?
3624	017110	001001			BNE	19\$	:BRANCH IF YES
3625	017112	104043			ERROR	43	:TOP OF MEMORY DOESN'T MATCH SIZE JUMPER
3626	017114	005037	001264		19\$: CLR	CPUEXP	:NO CPU TRAPS EXPECTED FOR AWHILE

:TEST 32 PARITY REPORTING THRU THE MAP, MAIN MEMORY EVEN WORD:THIS TEST FORCES A PARITY ERROR IN MAIN MEMORY AT PHYSICAL
:ADDRESS 00040000 BY SETTING ONE BIT IN EACH BYTE OF THE
:WORD. THE TEST THEN FORCES THE EVEN WORD PARITY BITS TO
:ONES AND READS ADDRESS 00040000 THRU THE UNIBUS MAP. THEN
:IT CHECKS TO SEE THAT THE PARITY ABORT OCCURRED AND THAT THE
:CONDITION WAS CORRECT BEFORE GOING TO THE NEXT TEST.:ERRORS ARE REPORTED INLINE IN THE TEST CODE, AND THE PARITY
:VECTOR WILL TRAP TO LABEL 10\$ AT THE END OF THIS TEST.

TST32:

3644	017120				SCOPE		
3645	017120	000004			MOV	#TST33,NXTTST	:SAVE STARTING ADDRESS OF NEXT TEST
3646	017122	012737	017362	001324			:FOR ESCAPE ON PARITY ERRORS
3647					TBITO		:TURN OFF T BIT TRAPPING IF ON
3648	017130	104416			20\$: MOV	#23404,\$TMP4	:EXPECTED ERROR CONDITION
3649	017132	012737	023404	001200	CLR	PPARER	:CLEAR MEM ERROR REG STORAGE
3650	017140	005037	001274		MOV	#-1,@MEMERR	:CLEAR MEMORY ERROR REGISTER
3651	017144	012737	177777	177744	MOV	@#CONTRL,\$TMP5	:SAVE CONTROL REG TO RESTORE CACHE
3652	017152	013737	177746	001202	MOV	#14,@#CONTRL	:FORCE MISSES IN CACHE
3653	017160	012737	000014	177746	MOV	LOWEST,KIPAR6	:PUT UNIBUS ADDR OF MAP REG IN PAR6
3654	017166	013737	001240	172354	MOV	#40000,@LREGL	:LOWEST MAP REGISTER POINTS TO 8K
3655	017174	012777	040000	162042	CLR	@LREGU	:CLEAR UPPER BITS OF MAP REG
3656	017202	005077	162040		MOV	#MAINT,R1	:PUT ADDR OF MAINT REG IN R1
3657	017206	012701	177750				

```
3658 017212 012737 017236 001110      MOV      #1$, $LPERR      ;SET LOOP ON ERROR POINTER TO 1$
3659 017220 012700 140000      MOV      #140000, R0      ;SELECT KIPAR6, OFFSET 0
3660 017224 012710 020001      MOV      #20001, (R0)      ;LOAD ONE BIT IN EACH BYTE OF 40000
3661 017230 012737 017322 000114      MOV      #10$, CACHVEC      ;SET PARITY VECTOR TO 10$
3662 017236 012737 030000 177750 1$:  MOV      #30000, @#MAINT      ;FORCE EVEN WORD PARITY BITS TO ONE
3663 017244 000240      NOP      ;THIS IS A SYNC POINT FOR SCOPING
3664 017246 011003      MOV      (R0), R3      ;'DATA FETCH' SHOULD CAUSE PARITY ABORT
3665 017250 011102      MOV      (R1), R2      ;SAVE MAINT REG IN CASE NO TRAP
3666 017252 005011      CLR      (R1)      ;CLEAR MAINT REG IN CASE NO TRAP
3667 017254 050000      BIS      R0, R0      ;DUMMY INST WITH P.B.'S ON
3668 017256 023737 001200 001274      CMP      $TMP4, PPARER      ;SEE IF ERROR REG HOLDS RIGHT DATA
3669 017264 001401      BEQ      2$      ;BRANCH IF CONDITION WAS CORRECT
3670 017266 104044      ERROR      44      ;PARITY REPORTING BAD
3671 017270 012737 177777 177744 2$:  MOV      #-1, MEMERR      ;CLEAR MEMORY ERROR REGISTER
3672 017276 012737 005764 000114      MOV      #MEMERR, CACHVEC      ;RESTORE PARITY SERVICE ROUTINE
3673 017304 013737 001202 177746      MOV      $TMP5, CONTRL      ;RESTORE CACHE TO FORMER CONDITION
3674 017312 012737 017132 001110      MOV      #20$, $LPERR      ;SET LOOP POINTER TO START OF TEST
3675 017320 000420      BR      TST33      ;:TEST OVER GO TO NEXT TEST
3676
3677
3678 017322 013737 177740 001270 10$:  MOV      @#LOADRS, PLOADR      ;SAVE LOWER CACHE ADDR REG
3679 017330 013737 177742 001272      MOV      @#HIADRS, PHIADR      ;SAVE HIGH BITS OF FAILING ADDR
3680 017336 013737 177744 001274      MOV      @#MEMERR, PPARER      ;SAVE MEMORY ERROR REGISTER
3681 017344 013737 177746 001276      MOV      @#CONTRL, PCONTR      ;SAVE CONTROL REGISTER FOR TYPE OUT
3682 017352 012737 030000 001300      MOV      #30000, PMAINT      ;SAVE DATA IN MAINTENANCE REGISTER
3683 017360 000002      RTI      ;RETURN TO TEST AND CHECK PARITY
3684
3685
3686
3687
3688
3689
3690
3691
3692
3693
3694
3695
3696
3697
3698
3699
3700 017362
3701 017362 000004
3702 017364 012737 017634 001324      SCOPE
3703      MOV      #TST34, NXTTST      ;SAVE STARTING ADDRESS OF NEXT TEST
3704 017372 012737 023410 001200 20$:  MOV      #23410, $TMP4      ;FOR ESCAPE ON PARITY ERRORS
3705 017400 005037 001274      CLR      PPARER      ;LOAD EXPECTED ERROR CONDITION
3706 017404 012737 177777 177744      MOV      #-1, @#MEMERR      ;CLEAR MEM ERROR REG STORAGE
3707 017412 013737 177746 001202      MOV      @#CONTRL, $TMP5      ;CLEAR MEMORY ERROR REGISTER
3708 017420 012737 000014 177746      MOV      #14, @#CONTRL      ;SAVE CONTROL REG TO RESTORE CACHE
3709 017426 013737 001240 172354      MOV      @#CONTRL, $TMP5      ;FORCE MISSES IN CACHE
3710 017434 012777 040000 161602      MOV      LOWEST, KIPAR6      ;PUT UNIBUS ADDR OF MA. REG IN PAR6
3711 017442 005077 161600      MOV      #40000, @LREGL      ;LOWEST MAP REGISTER POINTS TO 8K
3712 017446 012701 177750      CLR      @LREGU      ;CLEAR UPPER BITS OF MAP REG
3713 017452 012737 017510 001110      MOV      #MAINT, R1      ;PUT ADDR OF MAINT REG IN R1
3713      MOV      #1$, $LPERR      ;SET LOOP ON EROR POINTER TO 1$
```

```
*****
*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:
```


3714	017460	005037	140000		CLR	140000	:CLEAR LOCATION 40000
3715	017464	012700	140002		MOV	#140002,R0	:LOAD ADDRESS OF 40002 INTO R0
3716	017470	012737	017510	001110	MOV	#1\$,SLPERR	:SET LOOP ON ERROR POINTER TO 1\$
3717	017476	012710	020001		MOV	#20001,(R0)	:LOAD ONE BIT IN EACH BYTE OF 40002
3718	017502	012737	017574	000114	MOV	#10\$,CACHVEC	:SET PARITY VECTOR TO 10\$
3719	017510	012737	140000	177750	MOV	#140000,@#MAINT	:FORCE ODD WORD PARITY BITS TO ONE
3720	017516	000240			NOP		:THIS IS A SYNC POINT FOR SCOPING
3721	017520	011003			MOV	(R0),R3	: 'DATA FETCH' SHOULD CAUSE PARITY ABORT
3722	017522	011102			MOV	(R1),R2	:SAVE MAINT REG IN CASE NO TRAP
3723	017524	005011			CLR	(R1)	:CLEAR MAINT REG IN CASE NO TRAP
3724	017526	050000			BIS	R0,R0	:DUMMY INST WITH P.B.'S ON
3725	017530	023737	001200	001274	CMP	\$TMP4,PPARER	:SEE IF MEM ERROR REG HOLDS RIGHT DATA
3726	017536	001401			BEQ	2\$	:BRANCH IF ERROR WAS EXPECTED ONE
3727	017540	104044			ERROR	44	:PARITY REPORTING BAD
3728	017542	012737	177777	177744	MOV	#-1,MEMERR	:CLEAR MEMORY ERROR REGISTER
3729	017550	012737	005764	000114	MOV	#MEMERR,CACHVEC	:RESTORE PARITY SERVICE ROUTINE
3730	017556	013737	001202	177746	MOV	\$TMP5,CONTRL	:RESTORE CACHE TO FORMER CONDITION
3731	017564	012737	017372	001110	MOV	#20\$,SLPERR	:SET LOOP POINTER TO START OF TEST
3732	017572	000420			BR	TST34	:;TEST IS OVER, GO TO NEXT TEST
3733							
3734							
3735	017574	013737	177740	001270	MOV	@#LOADRS,PLOADR	:SAVE LOWER CACHE ADDR REG
3736	017602	013737	177742	001272	MOV	@#HIADRS,PHIADR	:SAVE HIGH BITS OF FAILING ADDR
3737	017610	013737	177744	001274	MOV	@#MEMERR,PPARER	:SAVE MEMORY ERROR REGISTER
3738	017616	013737	177746	001276	MOV	@#CONTRL,PCONTR	:SAVE CONTROL REGISTER FOR TYPE OUT
3739	017624	012737	140000	001300	MOV	#140000,PMAINT	:SAVE DATA IN MAINTENANCE REGISTER
3740	017632	000002			RTI		:RETURN TO TEST AND CHECK PARITY
3741							
3742							
3743							
3744							
3745							
3746							
3747							
3748							
3749							
3750							
3751							
3752							
3753							
3754							
3755							
3756							
3757							
3758	017634						
3759	017634	000004					
3760	017636	012737	020122	001324	MOV	#TST35,NXTTST	:SAVE STARTING ADDRESS OF NEXT TEST
3761							:FOR ESCAPE ON PARITY ERRORS
3762	017644	012737	177777	020010	MOV	#-1,21\$	:RESTORE NEG ONE FOR NEXT ITERATION
3763	017652	032737	000004	177746	BIT	#BIT2,@#CONTRL	:SEE IF GROUP 0 IS DISABLED
3764	017660	001120			BNE	TST35	:;BRANCH IF GROUP 0 IS DISABLED
3765	017662	013737	177746	001202	MOV	CONTRL,\$TMP5	:SAVE CONDITION OF CACHE
3766	017670	012737	023504	001200	MOV	#23504,\$TMP4	:EXPECTED PARITY CONDITION GROUP 0
3767	017676	005037	001274		CLR	PPARER	:CLEAR MEM ERROR REG STORAGE
3768	017702	012737	177777	177744	MOV	#-1,@#MEMERR	:CLEAR MEMORY ERROR REGISTER
3769	017710	013737	001240	172354	MOV	LOWEST,KIPAR6	:PUT UNIBUS ADDR OF MAP REG IN PAR6

:TEST 34 PARITY REPORTING THRU THE MAP, CACHE DATA GROUP 0
:*****
: THIS TEST FORCES A PARITY ERROR IN CACHE GROUP 0 AT PHYSICAL
: ADDRESS 00040010 BY SETTING BIT 15 IN THE WORD. THE TEST
: THEN FORCES GROUP 0 LOW BYTE PARITY BIT TO ZERO AND EVEN
: WORD HIGH BYTE PARITY BIT TO ONE BEFORE READING ADDRESS 00040010
: THRU THE UNIBUS MAP. THEN IT CHECKS TO SEE THAT THE PARITY
: ABORT OCCURRED AND THAT THE CONDITION WAS CORRECT BEFORE
: GOING TO THE NEXT TEST.
: ERRORS ARE REPORTED INLINE IN THE TEST CODE, AND THE PARITY
: VECTOR WILL TRAP TO LABEL 10\$ AT THE END OF THIS TEST.
:*****

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,@#CONTRL	:SEE IF GROUP 0 IS DISABLED
BNE	TST35	:;BRANCH IF GROUP 0 IS DISABLED
MOV	CONTRL,\$TMP5	:SAVE CONDITION OF CACHE
MOV	#23504,\$TMP4	:EXPECTED PARITY CONDITION GROUP 0
CLR	PPARER	:CLEAR MEM ERROR REG STORAGE
MOV	#-1,@#MEMERR	:CLEAR MEMORY ERROR REGISTER
MOV	LOWEST,KIPAR6	:PUT UNIBUS ADDR OF MAP REG IN PAR6

3770	017716	005077	161324		CLR	@LREGU	:CLEAR UPPER BITS ON MAP REG
3771	017722	012777	040010	161314	MOV	#40010,@LREGL	:LOAD LOWER 16 BITS OF MAP REG
3772	017730	012705	177750		MOV	#MAINT,R5	:PUT ADDR OF MAINT REG IN R5
3773	017734	012704	140000		MOV	#140000,R4	:SELECT PAR 6 BASE 0
3774	017740	012714	100000		MOV	#BIT15,(R4)	:MAKE SURE I GET SOFT AND HARD ERRORS
3775							:BY LOADING 40000 WITH BIT 15, THIS
3776							:CAUSES THE HIGH BYTE P.B. TO BE ON
3777							:AND THE LOW BYTE TO BE OFF
3778	017744	012737	000030	177746	MOV	#30,@#CONTRL	:FORCE SELECTION OF GROUP 0 ON READ
3779	017752	011401			MOV	(R4),R1	:PUT (40000) & (40002) IN GROUP 0
3780	017754	005001			CLR	R1	:ZERO R1 FOR THE MOMENT
3781	017756	012737	020062	000114	MOV	#10\$,CACHVEC	:SET PARITY VECTOR TO 10\$
3782	017764	010137	177750		MOV	R1,@#MAINT	:FORCE GPO PARITY BITS TO 0 ON 2ND PASS
3783	017770	000241			CLC		:THIS IS A SYNC POINT FOR SCOPING
3784							:THE LOW BYTE HAS ITS P.B. OFF
3785	017772	011401			MOV	(R4),R1	:DATA FETCH SHOULD CAUSE PARITY ABORT
3786	017774	011503			MOV	(R5),R3	:SAVE MAINT REG IN CASE NO TRAP
3787	017776	105015			CLRB	(R5)	:CLEAR MAINT REG IN CASE NO TRAP
3788	020000	006701			SXT	R1	:DUMMY INST WITH P.B.'S OFF
3789	020002	012701	020020		MOV	#020020,R1	:FORCE GROUP 0 LOW BYTE PARITY BIT
3790							:TO 0 AND MAIN MEMORY EVEN WORD HIGH
3791							:BYTE PARITY BIT TO 1
3792	020006	005227			INC	(PC)+	:INCREMENT NEXT WORD TO ZERO ON FIRST PASS
3793	020010	177777			.WORD	-1	:
3794	020012	001764			BEQ	1\$	:BRANCH ON FIRST PASS ONLY
3795	020014	023737	001200	001274	CMP	\$TMP4,PPARER	:SEE IF ERROR CONDITION MATCHES EXPECTED
3796	020022	001402			BEQ	2\$	:BRANCH IF CORRECT CONDITION
3797	020024	010302			MOV	R3,R2	:R2 HOLDS MAINT REG FOR ERROR TYPE OUT
3798	020026	104044			ERROR	44	:PARITY REPORTING BAD
3799	020030	012737	177777	177744	MOV	#-1,MEMERR	:CLEAR MEMORY ERROR REGISTER
3800	020036	012737	005764	000114	MOV	#MEMER,CACHVEC	:RESTORE PARITY SERVICE ROUTINE
3801	020044	013737	001202	177746	MOV	\$TMP5,CONTRL	:RESTORE CACHE TO FORMER CONDITION
3802	020052	012737	017644	001110	MOV	#20\$,LPERR	:SET LOOP POINTER TO START OF TEST
3803	020060	000420			BR	TST35	:TEST OVER GO TO NEXT TEST
3804							
3805							
3806	020062	013737	177740	001270	MOV	@#LOADRS,PLOADR	:SAVE LOWER CACHE ADDR REG
3807	020070	013737	177742	001272	MOV	@#HIADRS,PHIADR	:SAVE HIGH BITS OF FAILING ADDR
3808	020076	013737	177744	001274	MOV	@#MEMERR,PPARER	:SAVE MEMORY ERROR REGISTER
3809	020104	013737	177746	001276	MOV	@#CONTRL,PCONTR	:SAVE CONTROL REGISTER FOR TYPE OUT
3810	020112	012737	030060	001300	MOV	#30060,PMAINT	:SAVE DATA IN MAINTENANCE REGISTER
3811	020120	000002			RTI		:RETURN TO TEST AND CHECK PARITY
3812							
3813							
3814							
3815							
3816							
3817							
3818							
3819							
3820							
3821							
3822							
3823							
3824							
3825							

```
::*****
::*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
```



```
3826      ;*      VECTOR WILL TRAP TO LABEL 10$ AT THE END OF THIS TEST.
3827      ;*
3828      ;*****
3829      TST35:
3830      SCOPE
3831      MOV      #TST36,NXTTST      ;SAVE STARTING ADDRESS OF NEXT TEST
3832      ;FOR ESCAPE ON PARITY ERRORS
3833      20$:      MOV      #-1,21$      ;RESTORE NEG ONE FOR NEXT ITERATION
3834      BIT      #BIT3,CONTRL      ;SEE IF GROUP 1 IS DISABLED
3835      BNE      UBMAP      ;BRANCH TO NEXT TEST IF GROUP 1 DISABLED
3836      MOV      CONTRL,$TMP5      ;SAVE CONDITION OF CACHE
3837      MOV      #23604,$TMP4      ;EXPECTED PARITY CONDITION GROUP 1
3838      CLR      PPARER      ;CLEAR MEM ERROR REG STORAGE
3839      MOV      #-1,@MEMERR      ;CLEAR MEMORY ERROR REGISTER
3840      MOV      LOWEST,KIPAR6      ;PUT UNIBUS ADDR OF MAP REG IN PAR6
3841      CLR      @LREGU      ;CLEAR UPPER BITS ON MAP REG
3842      MOV      #40000,@LREGL      ;LOAD LOWER 16 BITS OF MAP REG
3843      MOV      #MAINT,R5      ;PUT ADDR OF MAINT REG IN R5
3844      MOV      #140000,R4      ;SELECT PAR 6 BASE 0
3845      MOV      #BIT15,(R4)      ;MAKE SURE I GET SOFT AND HARD ERRORS
3846      ;BY LOADING 40000 WITH BIT 15, THIS
3847      ;CAUSES THE HIGH BYTE P.B. TO BE ON
3848      ;AND THE LOW BYTE TO BE OFF
3849      MOV      #44,@CONTRL      ;FORCE SELECTION OF GROUP 0 ON READ
3850      MOV      (R4),R1      ;PUT (40000) & (40002) IN GROUP 1
3851      CLR      R1      ;CLEAR R1 FOR MOMENT
3852      MOV      #10$,CACHVEC      ;SET PARITY VECTOR TO 10$
3853      1$:      MOV      R1,@MAINT      ;FORCE GP1 PARITY BITS TO 0 ON 2ND PASS
3854      CLC      ;THIS IS A SYNC POINT FOR SCOPING
3855      ;THE LOW BYTE HAS ITS P.B. OFF
3856      MOV      (R4),R1      ;DATA FETCH SHOULD CAUSE PARITY ABORT
3857      MOV      (R5),R3      ;SAVE MAINT REG IN CASE NO TRAP
3858      CLRB      (R5)      ;CLEAR MAINT REG IN CASE NO TRAP
3859      SXT      R1      ;DUMMY INST WITH P.B.'S OFF
3860      MOV      #020100,R1      ;FORCE GROUP 1 LOW BYTE PARITY BIT
3861      ;TO 0 AND MAIN MEMORY EVEN WORD HIGH
3862      ;BYTE PARITY BIT TO 1
3863      INC      (PC)+      ;INCREMENT NEXT WORD TO 0
3864      21$:      .WORD      -1
3865      BEQ      1$      ;BRANCH ON FIRST PASS ONLY
3866      CMP      $TMP4,PPARER      ;SEE IF ERROR CONDITION MATCHES EXPECTED
3867      BEQ      2$      ;BRANCH IF ERROR WAS EXPECTED ONE
3868      MOV      R3,R2      ;R2 HOLDS MAINT REG FOR ERROR TYPE OUT
3869      ERROR      44      ;PARITY REPORTING BAD
3870      2$:      MOV      #-1,MEMERR      ;CLEAR MEMORY ERROR REGISTER
3871      MOV      #MEMER,CACHVEC      ;RESTORE PARITY SERVICE ROUTINE
3872      MOV      $TMP5,CONTRL      ;RESTORE CACHE TO FORMER CONDITION
3873      MOV      #20$, $LPERR      ;SET LOOP POINTER TO START OF TEST
3874      BR      UBMAP      ;TEST OVER GO TO NEXT TEST
3875
3876      10$:      MOV      @#LOADRS,PLOADR      ;SAVE LOWER CACHE ADDR REG
3877      MOV      @#HIADRS,PHIADR      ;SAVE HIGH BITS OF FAILING ADDR
3878      MOV      @#MEMERR,PPARER      ;SAVE MEMORY ERROR REGISTER
3879      MOV      @#CONTRL,PCONTR      ;SAVE CONTROL REGISTER FOR TYPE OUT
3880      MOV      #30300,PMAINT      ;SAVE DATA IN MAINTENANCE REGISTER
3881
```

```
3882 020406 000002 RTI ;RETURN TO TEST AND CHECK PARITY
3883
3884
3885
3886 .SBTTL *****
3887 .SBTTL THE FOLLOWING TESTS ARE RUN THROUGH THE UNIBUS MAP
3888 .SBTTL *****
3889
3890
3891 020410 000004 UBMAP: SCOPE ;LOOP ON PREVIOUS TEST
3892 020412 104420 TBTR ;RESTORE T-BIT IF IT WAS ON
3893
3894
3895
3896
3897
3898
3899
3900
3901
3902
3903
3904 020414 013701 001246 MOV LREGU,R1 ;PUT POINTER TO LOWEST MAP REG IN R1
3905 020420 005061 000004 CLR 4(R1) ;CLEAR UPPER 6 BITS OF (LOWEST + 1)
3906 020424 005061 000010 CLR 10(R1) ;CLEAR UPPER 6 BITS OF (LOWEST + 2)
3907 020430 012761 020000 000002 MOV #20000,2(R1) ;LOAD LOWER 16 BITS OF (LOWEST + 1)
3908 ;SO THAT IT POINTS TO 4K - 8K
3909 020436 005061 000006 CLR 6(R1) ;CLEAR LOWER 16 BITS OF (LOWEST + 2)
3910 020442 013701 001240 MOV LOWEST,R1 ;PREPARE TO LOAD PAR1
3911 020446 062701 000200 ADD #200,R1 ;POINTER TO (LOWEST + 1)
3912 020452 010137 172342 MOV R1,KIPAR1 ;LOAD PAR1 TO POINT TO LOWEST + 1
3913 020456 062701 000200 ADD #200,R1 ;ADJUST R1 FOR KIPAR0
3914 020462 010137 172340 MOV R1,KIPAR0 ;LOAD PAR0 TO POINT TO (LOWEST + 2)
3915
3916
3917
3918
3919
3920
3921
3922
3923
3924
3925
3926
3927
3928
3929 020466 TST36: *****
3930 020466 000004 SCOPE
3931 020470 012737 020526 001106 MOV #20$, $LPADR ;SET LOOP ON TEST POINTER TO 20$
3932 020476 012737 020526 001110 MOV #20$, $LPERR ;SET LOOP ON ERROR POINTER TO 20$
3933 020504 012737 000036 001102 MOV #36, $TSTNM ;SETUP TEST NUMBER AND CLR ERROR FLAG
3934 020512 013737 001102 177570 MOV $TSTNM,DISPLAY ;DISPLAY TEST NUMBER FOR ALL TO SEE
3935 020520 012737 020622 001324 MOV #TST37,NXTTST ;SET UP ESCAPE VECTOR IN CASE OF PARITY ERRORS
3936 020526 012737 000020 001264 20$: MOV #20,CPUEXP ;EXPECTING CPU TIMEOUT IN THIS TEST
3937 020534 013737 001240 172350 MOV LOWEST,KIPAR4 ;LOAD PAR 4 WITH LOWEST USABLE MAP REG
```



```
3938 020542 012777 000074 160476      MOV    #74,ALREGU      ;LOAD UPPER 6 BITS OF LOWEST MAP REG
3939 020550 012777 000000 160466      MOV    #000000,ALREGL ;LOAD LOWER 16 BITS OF LOWEST MAP REG
3940 020556 012737 020564 001110      MOV    #1$,SLPERR     ;SET LOOP ON ERROR POINTER TO 1$
3941 020564 005037 001266      1$:    CLR    PCPUER      ;CPU ERROR REG LOCATION
3942 020570 000240      NOP      ;THIS IS A SYNC POINT FOR SCOPING
3943 020572 013703 100000      MOV    @#100000,R3    ;TRY TO READ THRU PAGE 4
3944      ;THIS REFERENCE WILL GO OUT ON THE
3945      ;UNIBUS TO SELECT THE LOWEST USEABLE
3946      ;MAP REGISTER (DEFAULT MAP REG. 0).
3947      ;PHYSICAL ADDRESS 17700000 IS THEN
3948      ;GENERATED, WHICH SHOULD TIME OUT SINCE
3949      ;IT IS THE FIRST NON-EXISTANT LOCATION.
3950 020576 022737 000020 001266      CMP    #20,PCPUER     ;THE UNIBUS SHOULD HAVE TIMED OUT
3951 020604 001401      BEQ    10$      ;BRANCH IF UNIBUS TIMED OUT
3952 020606 104045      ERROR    45      ;DID NOT TIME OUT OVER UNIBUS
3953 020610 012737 020526 001110      10$:  MOV    #20$,SLPERR    ;SET LOOP POINTER TO START OF TEST
3954 020616 005037 001264      CLR    CPUEXP      ;NO CPU TRAPS EXPECTED IN NEXT TEST
3955
3956
3957
3958
3959
3960
3961
3962
3963
3964
3965
3966
3967
3968
3969
3970
```

```
::*****
:*TEST 37      RELOCATION TEST USING LOWEST USABLE MAPPING REG
:*
:*      THIS TEST CHECKS OUT THE FULL ADDITION PROPERTIES OF THE UNIBUS
:*      MAP A.L.U.. IN THE DEFAULT CASE IT USES MAP REGISTER ZERO BUT
:*      IF THE MAP JUMPERS HAVE BEEN ALTERED TO DE-SELECT SOME MAP REGISTERS
:*      THIS TEST WILL USE THE LOWEST USEABLE MAP REGISTER.
:*      IF AN ERROR OCCURS THE TEST WILL REPORT THE PHYSICAL ADDRESS
:*      THAT WAS DESIRED, AND THE DATA AT THE ADDRESS THAT WAS REFERENCED.
:*
:*      THIS TEST IS BEING RUN WITH ALL MEMORY REFERENCES GOING THRU
:*      THE UNIBUS MAP.
:*
::*****
```

```
3971 020622      TST37:
3972 020622 000004      SCOPE
3973 020624 012737 021536 001324      MOV    #TST40,NXTTST ;SAVE STARTING ADDRESS OF NEXT TEST
3974      ;FOR ESCAPE ON PARITY ERRORS
3975 020632 012737 060000 060000      MOV    #060000,@#060000 ;MAKE SURE THAT ADDRESS 060000
3976      ;CONTAINS ITS OWN ADDRESS AS DATA
3977
3978      ::
3979      ;*THE RELOCATION HERE USES A BASE OF 00060000 AND AN
3980      ;*OFFSET OF 00000 TO PRODUCE AN ADDRESS OF 00060000
3981 020640 012737 020672 001110      100$: MOV    #1$,SLPERR     ;SET LOOP ON ERROR POINTER TO 1$
3982 020646 005077 160374      CLR    ALREGU      ;CLEAR UPPER BITS OF MAPPING REG
3983 020652 012777 060000 160364      MOV    #060000,ALREGL ;LOAD LOWER BITS OF MAPPING REG
3984 020660 013737 001240 172354      MOV    LOWEST,@#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
3985 020666 012700 140000      MOV    #140000,R0      ;SELECT PAR6, OFFSET IS 00000
3986 020672 000240      1$:    NOP      ;THIS IS A SYNC POINT FOR SCOPING
3987 020674 011001      MOV    (R0),R1      ;READ LOCATION 060000 THRU THE UNIBUS
3988 020676 012702 060000      MOV    #060000,R2      ;THE EXPECTED PHYSICAL ADDRESS IS 060000
3989 020702 020102      CMP    R1,R2      ;SEE IF THE MAP'S FETCH WAS CORRECT
3990 020704 001401      BEQ    2$      ;BRANCH IF FETCHED DATA MATCHES ADDRESS
3991 020706 104046      ERROR    46      ;FAULTY RELOCATION BY UNIBUS MAP
3992
3993      ::
      ;*THE RELOCATION HERE USES A BASE OF 00052524 AND AN
```

```
3994 ;*OFFSET OF 05252 TO PRODUCE AN ADDRESS OF 00057776
3995 ::
3996 020710 012737 020742 001110 2$: MOV #3$, $LPERR ;SET LOOP ON ERROR POINTER TO 3$
3997 020716 005077 160324 CLR @LREGU ;CLEAR UPPER BITS OF MAPPING REG
3998 020722 012777 052524 160314 MOV #052524, @LREGL ;LOAD LOWER BITS OF MAPPING REG
3999 020730 013737 001240 172354 MOV LOWEST, @#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
4000 020736 012700 145252 MOV #145252, R0 ;SELECT PAR6, OFFSET IS 05252
4001 020742 000240 3$: NOP ;THIS IS A SYNC POINT FOR SCOPING
4002 020744 011001 MOV (R0), R1 ;READ LOCATION 057776 THRU THE UNIBUS
4003 020746 012702 057776 MOV #057776, R2 ;THE EXPECTED PHYSICAL ADDRESS IS 057776
4004 020752 020102 CMP R1, R2 ;SEE IF THE MAP'S FETCH WAS CORRECT
4005 020754 001401 BEQ 4$ ;BRANCH IF FETCHED DATA MATCHES ADDRESS
4006 020756 104046 ERROR 46 ;FAULTY RELOCATION BY UNIBUS MAP
4007 ::
4008 ;*THE RELOCATION HERE USES A BASE OF 00045252 AND AN
4009 ;*OFFSET OF 12524 TO PRODUCE AN ADDRESS OF 00057776
4010 ::
4011 020760 012737 021012 001110 4$: MOV #5$, $LPERR ;SET LOOP ON ERROR POINTER TO 5$
4012 020766 005077 160254 CLR @LREGU ;CLEAR UPPER BITS OF MAPPING REG
4013 020772 012777 045252 160244 MOV #045252, @LREGL ;LOAD LOWER BITS OF MAPPING REG
4014 021000 013737 001240 172354 MOV LOWEST, @#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
4015 021006 012700 152524 MOV #152524, R0 ;SELECT PAR6, OFFSET IS 12524
4016 021012 000240 5$: NOP ;THIS IS A SYNC POINT FOR SCOPING
4017 021014 011001 MOV (R0), R1 ;READ LOCATION 057776 THRU THE UNIBUS
4018 021016 012702 057776 MOV #057776, R2 ;THE EXPECTED PHYSICAL ADDRESS IS 057776
4019 021022 020102 CMP R1, R2 ;SEE IF THE MAP'S FETCH WAS CORRECT
4020 021024 001401 BEQ 6$ ;BRANCH IF FETCHED DATA MATCHES ADDRESS
4021 021026 104046 ERROR 46 ;FAULTY RELOCATION BY UNIBUS MAP
4022 ::
4023 ;*THE RELOCATION HERE USES A BASE OF 00050420 AND AN
4024 ;*OFFSET OF 10420 TO PRODUCE AN ADDRESS OF 00061040
4025 ::
4026 021030 012737 021062 001110 6$: MOV #7$, $LPERR ;SET LOOP ON ERROR POINTER TO 7$
4027 021036 005077 160204 CLR @LREGU ;CLEAR UPPER BITS OF MAPPING REG
4028 021042 012777 050420 160174 MOV #050420, @LREGL ;LOAD LOWER BITS OF MAPPING REG
4029 021050 013737 001240 172354 MOV LOWEST, @#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
4030 021056 012700 150420 MOV #150420, R0 ;SELECT PAR6, OFFSET IS 10420
4031 021062 000240 7$: NOP ;THIS IS A SYNC POINT FOR SCOPING
4032 021064 011001 MOV (R0), R1 ;READ LOCATION 061040 THRU THE UNIBUS
4033 021066 012702 061040 MOV #061040, R2 ;THE EXPECTED PHYSICAL ADDRESS IS 061040
4034 021072 020102 CMP R1, R2 ;SEE IF THE MAP'S FETCH WAS CORRECT
4035 021074 001401 BEQ 8$ ;BRANCH IF FETCHED DATA MATCHES ADDRESS
4036 021076 104046 ERROR 46 ;FAULTY RELOCATION BY UNIBUS MAP
4037 ::
4038 ;*THE RELOCATION HERE USES A BASE OF 00054630 AND AN
4039 ;*OFFSET OF 04210 TO PRODUCE AN ADDRESS OF 00061040
4040 ::
4041 021100 012737 021132 001110 8$: MOV #9$, $LPERR ;SET LOOP ON ERROR POINTER TO 9$
4042 021106 005077 160134 CLR @LREGU ;CLEAR UPPER BITS OF MAPPING REG
4043 021112 012777 054630 160124 MOV #054630, @LREGL ;LOAD LOWER BITS OF MAPPING REG
4044 021120 013737 001240 172354 MOV LOWEST, @#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
4045 021126 012700 144210 MOV #144210, R0 ;SELECT PAR6, OFFSET IS 04210
4046 021132 000240 9$: NOP ;THIS IS A SYNC POINT FOR SCOPING
4047 021134 011001 MOV (R0), R1 ;READ LOCATION 061040 THRU THE UNIBUS
4048 021136 012702 061040 MOV #061040, R2 ;THE EXPECTED PHYSICAL ADDRESS IS 061040
4049 021142 020102 CMP R1, R2 ;SEE IF THE MAP'S FETCH WAS CORRECT
```



```
4050 021144 001401      BEQ      10$      ;BRANCH IF FETCHED DATA MATCHES ADDRESS
4051 021146 104046      ERROR    46      ;FAULTY RELOCATION BY UNIBUS MAP
4052
4053      ;;
4054      ;*THE RELOCATION HERE USES A BASE OF 00044210 AND AN
4055      ;*OFFSET OF 14630 TO PRODUCE AN ADDRESS OF 00061040
4056 021150 012737 021202 001110 10$:  MOV      #11$, $LPERR      ;SET LOOP ON ERROR POINTER TO 11$
4057 021156 005077 160064      CLR      @LREGU      ;CLEAR UPPER BITS OF MAPPING REG
4058 021162 012777 044210 160054      MOV      #044210, @LREGL      ;LOAD LOWER BITS OF MAPPING REG
4059 021170 013737 001240 172354      MOV      LOWEST, @#KIPAR6      ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
4060 021176 012700 154630      MOV      #154630, R0      ;SELECT PAR6, OFFSET IS 14630
4061 021202 000240      11$:  NOP      ;THIS IS A SYNC POINT FOR SCOPING
4062 021204 011001      MOV      (R0), R1      ;READ LOCATION 061040 THRU THE UNIBUS
4063 021206 012702 061040      MOV      #061040, R2      ;THE EXPECTED PHYSICAL ADDRESS IS 061040
4064 021212 020102      CMP      R1, R2      ;SEE IF THE MAP'S FETCH WAS CORRECT
4065 021214 001401      BEQ      12$      ;BRANCH IF FETCHED DATA MATCHES ADDRESS
4066 021216 104046      ERROR    46      ;FAULTY RELOCATION BY UNIBUS MAP
4067
4068      ;;
4069      ;*THE RELOCATION HERE USES A BASE OF 00056734 AND AN
4070      ;*OFFSET OF 02104 TO PRODUCE AN ADDRESS OF 00061040
4071 021220 012737 021252 001110 12$:  MOV      #13$, $LPERR      ;SET LOOP ON ERROR POINTER TO 13$
4072 021226 005077 160014      CLR      @LREGU      ;CLEAR UPPER BITS OF MAPPING REG
4073 021232 012777 056734 160004      MOV      #056734, @LREGL      ;LOAD LOWER BITS OF MAPPING REG
4074 021240 013737 001240 172354      MOV      LOWEST, @#KIPAR6      ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
4075 021246 012700 142104      MOV      #142104, R0      ;SELECT PAR6, OFFSET IS 02104
4076 021252 000240      13$:  NOP      ;THIS IS A SYNC POINT FOR SCOPING
4077 021254 011001      MOV      (R0), R1      ;READ LOCATION 061040 THRU THE UNIBUS
4078 021256 012702 061040      MOV      #061040, R2      ;THE EXPECTED PHYSICAL ADDRESS IS 061040
4079 021262 020102      CMP      R1, R2      ;SEE IF THE MAP'S FETCH WAS CORRECT
4080 021264 001401      BEQ      14$      ;BRANCH IF FETCHED DATA MATCHES ADDRESS
4081 021266 104046      ERROR    46      ;FAULTY RELOCATION BY UNIBUS MAP
4082
4083      ;;
4084      ;*THE RELOCATION HERE USES A BASE OF 00042104 AND AN
4085      ;*OFFSET OF 16734 TO PRODUCE AN ADDRESS OF 00061040
4086 021270 012737 021322 001110 14$:  MOV      #15$, $LPERR      ;SET LOOP ON ERROR POINTER TO 15$
4087 021276 005077 157744      CLR      @LREGU      ;CLEAR UPPER BITS OF MAPPING REG
4088 021302 012777 042104 157734      MOV      #042104, @LREGL      ;LOAD LOWER BITS OF MAPPING REG
4089 021310 013737 001240 172354      MOV      LOWEST, @#KIPAR6      ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
4090 021316 012700 156734      MOV      #156734, R0      ;SELECT PAR6, OFFSET IS 16734
4091 021322 000240      15$:  NOP      ;THIS IS A SYNC POINT FOR SCOPING
4092 021324 011001      MOV      (R0), R1      ;READ LOCATION 061040 THRU THE UNIBUS
4093 021326 012702 061040      MOV      #061040, R2      ;THE EXPECTED PHYSICAL ADDRESS IS 061040
4094 021332 020102      CMP      R1, R2      ;SEE IF THE MAP'S FETCH WAS CORRECT
4095 021334 001401      BEQ      16$      ;BRANCH IF FETCHED DATA MATCHES ADDRESS
4096 021336 104046      ERROR    46      ;FAULTY RELOCATION BY UNIBUS MAP
4097
4098      ;;
4099      ;*THE RELOCATION HERE USES A BASE OF 00057776 AND AN
4100      ;*OFFSET OF 01042 TO PRODUCE AN ADDRESS OF 00061040
4101 021340 012737 021372 001110 16$:  MOV      #17$, $LPERR      ;SET LOOP ON ERROR POINTER TO 17$
4102 021346 005077 157674      CLR      @LREGU      ;CLEAR UPPER BITS OF MAPPING REG
4103 021352 012777 057776 157664      MOV      #057776, @LREGL      ;LOAD LOWER BITS OF MAPPING REG
4104 021360 013737 001240 172354      MOV      LOWEST, @#KIPAR6      ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
4105 021366 012700 141042      MOV      #141042, R0      ;SELECT PAR6, OFFSET IS 01042
```

```
4106 021372 000240      17$:  NOP      ;THIS IS A SYNC POINT FOR SCOPING
4107 021374 011001      MOV      (R0),R1 ;READ LOCATION 061040 THRU THE UNIBUS
4108 021376 012702 061040 MOV      #061040,R2 ;THE EXPECTED PHYSICAL ADDRESS IS 061040
4109 021402 020102      CMP      R1,R2 ;SEE IF THE MAP'S FETCH WAS CORRECT
4110 021404 001401      BEQ      18$ ;BRANCH IF FETCHED DATA MATCHES ADDRESS
4111 021406 104046      ERROR    46 ;FAULTY RELOCATION BY UNIBUS MAP
```

```
4112      ;;
4113      ;*THE RELOCATION HERE USES A BASE OF 00041042 AND AN
4114      ;*OFFSET OF 17776 TO PRODUCE AN ADDRESS OF 00061040
4115      ;*
```

```
4116 021410 012737 021442 001110 18$:  MOV      #19$,$LPERR ;SET LOOP ON ERROR POINTER TO 19$
4117 021416 005077 157624      CLR      @LREGU ;CLEAR UPPER BITS OF MAPPING REG
4118 021422 012777 041042 157614      MOV      #041042,@LREGL ;LOAD LOWER BITS OF MAPPING REG
4119 021430 013737 001240 172354      MOV      LOWEST,@#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
4120 021436 012700 157776      MOV      #157776,R0 ;SELECT PAR6, OFFSET IS 17776
4121 021442 000240      19$:  NOP      ;THIS IS A SYNC POINT FOR SCOPING
4122 021444 011001      MOV      (R0),R1 ;READ LOCATION 061040 THRU THE UNIBUS
4123 021446 012702 061040 MOV      #061040,R2 ;THE EXPECTED PHYSICAL ADDRESS IS 061040
4124 021452 020102      CMP      R1,R2 ;SEE IF THE MAP'S FETCH WAS CORRECT
4125 021454 001401      BEQ      20$ ;BRANCH IF FETCHED DATA MATCHES ADDRESS
4126 021456 104046      ERROR    46 ;FAULTY RELOCATION BY UNIBUS MAP
```

```
4127      ;;
4128      ;*THE RELOCATION HERE USES A BASE OF 00057776 AND AN
4129      ;*OFFSET OF 00002 TO PRODUCE AN ADDRESS OF 00060000
4130      ;*
```

```
4131 021460 012737 021512 001110 20$:  MOV      #21$,$LPERR ;SET LOOP ON ERROR POINTER TO 21$
4132 021466 005077 157554      CLR      @LREGU ;CLEAR UPPER BITS OF MAPPING REG
4133 021472 012777 057776 157544      MOV      #057776,@LREGL ;LOAD LOWER BITS OF MAPPING REG
4134 021500 013737 001240 172354      MOV      LOWEST,@#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
4135 021506 012700 140002      MOV      #140002,R0 ;SELECT PAR6, OFFSET IS 00002
4136 021512 000240      21$:  NOP      ;THIS IS A SYNC POINT FOR SCOPING
4137 021514 011001      MOV      (R0),R1 ;READ LOCATION 060000 THRU THE UNIBUS
4138 021516 012702 060000 MOV      #060000,R2 ;THE EXPECTED PHYSICAL ADDRESS IS 060000
4139 021522 020102      CMP      R1,R2 ;SEE IF THE MAP'S FETCH WAS CORRECT
4140 021524 001401      BEQ      22$ ;BRANCH IF FETCHED DATA MATCHES ADDRESS
4141 021526 104046      ERROR    46 ;FAULTY RELOCATION BY UNIBUS MAP
4142 021530 012737 020640 001110 22$:  MOV      #100$,$LPERR ;SET LOOP POINTER TO START OF TEST
```

```
4143      ;;
4144      ;*****
4145      ;*TEST 40      TEST CARRY PROPAGATION OF MAP'S RELOCATION ADDER
4146      ;*
4147      ;*      EVERY ADDRESS OF THE FORM XXXX0000 IS GENERATED HERE STARTING
4148      ;*      WITH 00030000 UP TO 17000000. THAT IS THE FIRST OF EVERY 2K
4149      ;*      WORDS IS ADDRESSED, TO INSURE THAT THE ADDER IN THE MAP IS
4150      ;*      WORKING PROPERLY AND, THE SYSTEM SIZE JUMPERS ARE ACTUALLY SET
4151      ;*      FOR THE TOP OF MAIN MEMORY.
4152      ;*
4153      ;*****
```

```
4154      ;*****
4155      ;*****
4156      ;*****
4157 021536      TST40:
4158 021536 000004      SCOPE
4159 021540 012737 022042 001324      MOV      #TST41,NXTTST ;SAVE STARTING ADDRESS OF NEXT TEST
4160      ;FOR ESCAPE ON PARITY ERRORS
4161 021546 012737 000012 001204      MOV      #12,$TIMES ;DO 12 ITERATIONS
```


4162	021554	012737	177777	021710	20\$:	MOV	#-1,4\$	:INITIALIZE FLAG AS NEGATIVE ONE
4163	021562	005077	157460			CLR	@LREGU	:CLEAR UPPER 6 BITS OF MAP REG
4164	021566	012777	020000	157450		MOV	#20000,@LREGL	:LOAD 4K BASE INTO MAP REGISTER
4165	021574	012701	100100			MOV	#100100,R1	:LOAD BITS TO SELECT PAR 4, OFFSET 100
4166	021600	012700	150000			MOV	#150000,R0	:LOAD BITS TO SELECT PAR 6, OFFSET 2K
4167	021604	012737	000277	172350		MOV	#277,KIPAR4	:START WITH PHYSICAL 6K
4168	021612	013737	001240	172354		MOV	LOWEST,KIPAR6	:LOAD PAR 6 WITH MAP REG'S ADDR
4169	021620	012737	021660	001110		MOV	#10\$,\$LPERR	:SET LOOP ON ERROR POINTER TO 10\$
4170	021626	012737	000020	001264	3\$:	MOV	#TIMOUT,CPUEXP	:EXPECTING A UNIBUS TIME OUT DURING TEST
4171	021634	005037	001266			CLR	PCPUER	:CLEAR TIME OUT FLAG
4172	021640	013710	001326			MOV	DATA,(R0)	:THIS LOAD WILL TIME OUT WHEN YOU
4173								:HAVE REACHED THE TOP OF MEMORY
4174								:IT SELECTS PAR 6 WHICH WILL PUT ADDR
4175								:<XXX XX1>0000 ON THE UNIBUS.
4176								:THE X'S WILL SELECT THE LOWEST USEABLE
4177								:MAPPING REGISTER. THE DEFAULT CASE IS
4178								:010000, SELECTING MAPPING REGISTER 0.
4179	021644	005737	001266			TST	PCPUER	:SEE IF THERE WAS MAIN MEMORY
4180	021650	001016				BNE	1\$	:BRANCH IF NO MAIN MEMORY FROM UNIBUS
4181	021652	012737	000040	001264		MOV	#NEXMEM,CPUEXP	:POSSIBLE CACHE NON-EXISTANT MEMORY
4182	021660	011103			10\$:	MOV	(R1),R3	:READ TEST LOCATION VIA FASTBUS
4183	021662	022737	000040	001266		CMP	#NEXMEM,PCPUER	:WAS THERE CACHE NON-EXISTANT MEMORY
4184	021670	001414				BEQ	2\$	:BRANCH IF NON-EXISTANT MEMORY
4185	021672	000240				NOP		:THIS IS A SYNC POINT FOR SCOPING
4186	021674	011002				MOV	(R0),R2	:READ TEST LOCATION VIA UNIBUS MAP
4187	021676	020203				CMP	R2,R3	:COMPARE TEST DATA R2=MAP DATA
4188								:R3=FASTBUS DATA
4189	021700	001410				BEQ	2\$	:BRANCH IF IT WAS THE SAME
4190	021702	104047				ERROR	47	:FAILURE IN CARRY PROPAGATION IN MAP
4191	021704	000406				BR	2\$	:BRANCH TO UPDATE ROUTINE
4192	021706	005227			1\$:	INC	(PC)+	:INCREMENT ONE TIME ENTRANCE FLAG
4193	021710	177777			4\$:	.WORD	-1	:USE NEGATIVE ONE FOR FLAG
4194	021712	001003				BNE	2\$	:BRANCH IF YOU'VE BEEN HERE BEFORE
4195	021714	013737	172350	001320		MOV	KIPAR4,RSIZE	:SAVE UPPER LIMIT OF MEMORY
4196	021722	062737	000100	001326	2\$:	ADD	#100,DATA	:CHANGE PATTERN FOR NEXT LOAD
4197	021730	062737	000100	172350		ADD	#100,KIPAR4	:ADD 2K TO PAR4
4198	021736	062777	010000	157300		ADD	#10000,@LREGL	:ADD 2K TO MAP REGISTER
4199	021744	001330				BNE	3\$	:BRANCH IF MAP REGISTER NOT ZERO
4200	021746	005277	157274			INC	@LREGU	:ADD ONE TO UPPER 6 BITS OF MAP REG
4201	021752	022777	000073	157266		CMP	#73,@LREGU	:SEE IF TOP 128K BLOCK HAS BEEN PASSED
4202	021760	103322				BHIS	3\$	:BRANCH IF NOT PAST IT
4203	021762	005237	001326			INC	DATA	:CHANGE DATA PATTERN FOR NEXT PASS
4204	021766	042737	177700	001326		BIC	#177700,DATA	:CLEAR UPPER 10 BITS OF DATA PATTERN
4205	021774	052737	000300	001326		BIS	#300,DATA	:START WITH 3XX IN DATA PATTERN
4206	022002	012737	021554	001110		MOV	#20\$,\$LPERR	:SET LOOP POINTER TO START OF TEST
4207	022010	023737	177760	001320		CMP	@#SIZELO,RSIZE	:SEE IF SIZE JUMPERS AGREE WITH MEMORY TOP
4208	022016	001407				BEQ	19\$	:BRANCH IF TOP OF MEMORY AGREES WITH
4209								:THE SIZE JUMPERS
4210	022020	105737	001331			TSTB	KB11EM	:RUNNING ON A KB11-EM?
4211	022024	001004				BNE	19\$	:SKIP ERROR IF YES
4212	022026	105737	001332			TSTB	KB11CM	:RUNNING ON MODIFIED MACHINE?
4213	022032	001001				BNE	19\$	:BRANCH IF YES
4214	022034	104050				ERROR	50	:MEMORY SIZE JUMPERS NOT RIGHT
4215	022036	005037	001264		19\$:	CLR	CPUEXP	:NO CPU TRAPS EXPECTED IN NEXT TEST
4216								
4217								

4218						::*****
4219						::TEST 41
4220						PARITY REPORTING THRU THE MAP, MAIN MEMORY EVEN WORD
4221						::*
4222						::*
4223						::*
4224						::*
4225						::*
4226						::*
4227						::*
4228						::*
4229						::*
4230						::*
4231						::*
4232						::*
4233	022042					TST41:
4234	022042	000004				SCOPE
4235	022044	012737	022304	001324		MOV #TST42,NXTTST ;SAVE STARTING ADDRESS OF NEXT TEST
4236						;FOR ESCAPE ON PARITY ERRORS
4237	022052	104416				;TURN OFF T BIT TRAPPING IF ON
4238	022054	012737	023404	001200	20\$:	TBITO
4239	022062	005037	001274			MOV #23404,\$TMP4 ;EXPECTED ERROR CONDITION
4240	022066	012737	177777	177744		CLR PPARER ;CLEAR MEM ERROR REG STORAGE
4241	022074	013737	177746	001202		MOV #-1,@MEMERR ;CLEAR MEMORY ERROR REGISTER
4242	022102	012737	000014	177746		MOV @WCONTRL,\$TMP5 ;SAVE CONTROL REG TO RESTORE CACHE
4243	022110	013737	001240	172354		MOV #14,@WCONTRL ;FORCE MISSES IN CACHE
4244	022116	012777	040000	157120		MOV LOWEST,KIPAR6 ;PUT UNIBUS ADDR OF MAP REG IN PAR6
4245	022124	005077	157116			MOV #40000,@LREGL ;LOWEST MAP REGISTER POINTS TO 8K
4246	022130	012701	177750			CLR @LREGU ;CLEAR UPPER BITS OF MAP REG
4247	022134	012737	022152	001110		MOV @MAINT,R1 ;PUT ADDR OF MAINT REG IN R1
4248	022142	012700	140000			MOV #1\$, \$LPERR ;SET LOOP ON ERROR POINTER TO 1\$
4249	022146	012710	020001			MOV #140000,R0 ;SELECT KIPAR6, OFFSET 0
4250	022152	012737	022244	000114	1\$:	MOV #20001,(R0) ;LOAD ONE BIT IN EACH BYTE OF 40000
4251	022160	012737	030000	177750		MOV #10\$,CACHVEC ;SET PARITY VECTOR TO 10\$
4252	022166	000240				MOV #30000,@MAINT ;FORCE EVEN WORD PARITY BITS TO ONE
4253	022170	011003				NOP ;THIS IS A SYNC POINT FOR SCOPING
4254	022172	011102				MOV (R0),R3 ;'DATA FETCH' SHOULD CAUSE PARITY ABORT
4255	022174	005011				MOV (R1),R2 ;SAVE MAINT REG IN CASE NO TRAP
4256	022176	050000				CLR (R1) ;CLEAR MAINT REG IN CASE NO TRAP
4257	022200	023737	001200	001274		BIS R0,R0 ;DUMMY INST WITH P.B.'S ON
4258	022206	001401				CMP \$TMP4,PPARER ;SEE IF ERROR REG HOLDS RIGHT DATA
4259	022210	104051				BEQ 2\$;BRANCH IF CONDITION WAS CORRECT
4260	022212	012737	177777	177744	2\$:	ERROR 51 ;FAILURE IN PARITY REPORTING
4261	022220	012737	005764	000114		MOV #-1,MEMERR ;CLEAR MEMORY ERROR REGISTER
4262	022226	013737	001202	177746		MOV @MEMERR,CACHVEC ;RESTORE PARITY SERVICE ROUTINE
4263	022234	012737	022054	001110		MOV \$TMP5,CONTRL ;RESTORE CACHE TO FORMER CONDITION
4264	022242	000420				MOV #20\$, \$LPERR ;SET LOOP POINTER TO START OF TEST
4265						BR TST42 ;;TEST OVER GO TO NEXT TEST
4266						
4267	022244	013737	177740	001270	10\$:	MOV @WLOADRS,PLOADR ;SAVE LOWER CACHE ADDR REG
4268	022252	013737	177742	001272		MOV @WHIADRS,PHIADR ;SAVE HIGH BITS OF FAILING ADDR
4269	022260	013737	177744	001274		MOV @WMEMERR,PPARER ;SAVE MEMORY ERROR REGISTER
4270	022266	013737	177746	001276		MOV @WCONTRL,PCONTR ;SAVE CONTROL REGISTER FOR TYPE OUT
4271	022274	012737	030000	001300		MOV #30000,PMAINT ;SAVE DATA IN MAINTENANCE REGISTER
4272	022302	000002				RTI ;RETURN TO TEST AND CHECK PARITY
4273						

4274
4275
4276
4277
4278
4279
4280
4281
4282
4283
4284
4285
4286
4287
4288
4289
4290
4291*****
:TEST 42 PARITY REPORTING THRU THE MAP, MAIN MEMORY ODD WORD:THIS TEST FORCES A PARITY ERROR IN MAIN MEMORY AT PHYSICAL
:ADDRESS 00040002 BY SETTING ONE BIT IN EACH BYTE OF THE
:WORD. THE TEST THEN FORCES THE ODD WORD PARITY BITS TO
:ONES AND READS ADDRESS 00040002 THRU THE UNIBUS MAP. THEN
:IT CHECKS TO SEE THAT THE PARITY ABORT OCCURRED AND THAT THE
:CONDITION WAS CORRECT BEFORE GOING TO THE NEXT TEST.:ERRORS ARE REPORTED INLINE IN THE TEST CODE, AND THE PARITY
:VECTOR WILL TRAP TO LABEL 10\$ AT THE END OF THIS TEST.:THIS TEST IS BEING RUN WITH ALL MEMORY REFERENCES GOING THRU
:THE UNIBUS MAP.*****
TST42:

4292	022304					SCOPE		
4293	022304	000004				MOV	#TST43,NXTTST	:SAVE STARTING ADDRESS OF NEXT TEST
4294	022306	012737	022556	001324				:FOR ESCAPE ON PARITY ERRORS
4295								:LOAD EXPECTED ERROR CONDITION
4296	022314	012737	023410	001200	20\$:	MOV	#23410,\$TMP4	:CLEAR MEM ERROR REG STORAGE
4297	022322	005037	001274			CLR	PPARER	:CLEAR MEMORY ERROR REGISTER
4298	022326	012737	177777	177744		MOV	#-1,@MEMERR	:SAVE CONTROL REG TO RESTORE CACHE
4299	022334	013737	177746	001202		MOV	@#CONTRL,\$TMP5	:FORCE MISSES IN CACHE
4300	022342	012737	000014	177746		MOV	#14,@#CONTRL	:PUT UNIBUS ADDR OF MAP REG IN PAR6
4301	022350	013737	001240	172354		MOV	LOWEST,KIPAR6	:LOWEST MAP REGISTER POINTS TO 8K
4302	022356	012777	040000	156660		MOV	#40000,@LREG	:CLEAR UPPER BITS OF MAP REG
4303	022364	005077	156656			CLR	@LREG	:PUT ADDR OF MAINT REG IN R1
4304	022370	012701	177750			MOV	#MAINT,R1	:SET LOOP ON ERROR POINTER TO 1\$
4305	022374	012737	022424	001110		MOV	#1\$,\$LPERR	:CLEAR LOCATION 40000
4306	022402	005037	140000			CLR	140000	:LOAD ADDRESS OF 40002 INTO R0
4307	022406	012700	140002			MOV	#140002,R0	:SET LOOP ON ERROR POINTER TO 1\$
4308	022412	012737	022424	001110		MOV	#1\$,\$LPERR	:LOAD ONE BIT IN EACH BYTE OF 40002
4309	022420	012710	020001			MOV	#20001,(R0)	:SET PARITY VECTOR TO 10\$
4310	022424	012737	022516	000114	1\$:	MOV	#10\$,CACHVEC	:FORCE ODD WORD PARITY BITS TO ONE
4311	022432	012737	140000	177750		MOV	#140000,@MAINT	:THIS IS A SYNC POINT FOR SCOPING
4312	022440	000240				NOP		:DATA FETCH SHOULD CAUSE PARITY ABORT
4313	022442	011003				MOV	(R0),R3	:SAVE MAINT REG IN CASE NO TRAP
4314	022444	011102				MOV	(R1),R2	:CLEAR MAINT REG IN CASE NO TRAP
4315	022446	005011				CLR	(R1)	:DUMMY INST WITH P.B.'S ON
4316	022450	050000				BIS	R0,R0	:SEE IF MEM ERROR REG HOLDS RIGHT DATA
4317	022452	023737	001200	001274		CMP	\$TMP4,PPARER	:BRANCH IF ERROR WAS EXPECTED ONE
4318	022460	001401				BEQ	2\$	:FAILURE IN PARITY REPORTING
4319	022462	104051				ERROR	51	:CLEAR MEMORY ERROR REGISTER
4320	022464	012737	177777	177744	2\$:	MOV	#-1,MEMERR	:RESTORE PARITY SERVICE ROUTINE
4321	022472	012737	005764	000114		MOV	#MEMER,CACHVEC	:RESTORE CACHE TO FORMER CONDITION
4322	022500	013737	001202	177746		MOV	\$TMP5,CONTRL	:SET LOOP POINTER TO START OF TEST
4323	022506	012737	022314	001110		MOV	#20\$,\$LPERR	:TEST IS OVER, GO TO NEXT TEST
4324	022514	000420				BR	TST43	
4325								
4326								
4327	022516	013737	177740	001270	10\$:	MOV	@#LOADRS,PLOADR	:SAVE LOWER CACHE ADDR REG
4328	022524	013737	177742	001272		MOV	@#HIADRS,PHIADR	:SAVE HIGH BITS OF FAILING ADDR
4329	022532	013737	177744	001274		MOV	@#MEMERR,PPARER	:SAVE MEMORY ERROR REGISTER

4330 022540 013737 177746 001276
4331 022546 012737 140000 001300
4332 022554 000002

MOV @#CONTRL,PCONTR ;SAVE CONTROL REGISTER FOR TYPE OUT
MOV #140000,PMaint ;SAVE DATA IN MAINTENANCE REGISTER
RTI ;RETURN TO TEST AND CHECK PARITY

4333
4334
4335
4336
4337
4338
4339
4340
4341
4342
4343
4344
4345
4346
4347
4348
4349
4350
4351
4352

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

:
: THIS TEST FORCES A PARITY ERROR IN CACHE GROUP 0 AT PHYSICAL
: ADDRESS 00040000 BY SETTING BIT 15 IN THE WORD. THE TEST
: THEN FORCES GROUP 0 LOW BYTE PARITY BIT TO ZERO AND EVEN
: WORD HIGH BYTE PARITY BIT TO ONE BEFORE READING ADDRESS 00040000
: THRU THE UNIBUS MAP. THEN IT CHECKS TO SEE THAT THE PARITY
: ABORT OCCURRED AND THAT THE CONDITION WAS CORRECT BEFORE
: GOING TO THE NEXT TEST.

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

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

:TST43:

4353 022556
4354 022556 000004
4355 022560 012737 023044 001324
4356
4357 022566 012737 177777 022732
4358 022574 032737 000004 177746
4359 022602 001120
4360 022604 013737 177746 001202
4361 022612 012737 023504 001200
4362 022620 005037 001274
4363 022624 012737 177777 177744
4364 022632 013737 001240 172354
4365 022640 005077 156402
4366 022644 012777 040000 156372
4367 022652 012705 177750
4368 022656 012704 140000
4369 022662 012714 100000

SCOPE
MOV #TST44,NXTTST ;SAVE STARTING ADDRESS OF NEXT TEST
;FOR ESCAPE ON PARITY ERRORS
20\$: MOV #-1,21\$;RESTORE NEG ONE FOR NEXT ITERATION
BIT #BIT2,@#CONTRL ;SEE IF GROUP 0 IS DISABLED
BNE TST44 ;BRANCH IF GROUP 0 IS DISABLED
MOV CONTRL,\$TMP5 ;SAVE CONDITION OF CACHE
MOV #23504,\$TMP4 ;EXPECTED PARITY CONDITION GROUP 0
CLR PPARER ;CLEAR MEM ERROR REG STORAGE
MOV #-1,@#MEMERR ;CLEAR MEMORY ERROR REGISTER
MOV LOWEST,KIPAR6 ;PUT UNIBUS ADDR OF MAP REG IN 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
;FORCE SELECTION OF GROUP 0 ON READ
;PUT (40000) & (40002) IN GROUP 0
;ZERO R1 FOR THE MOMENT
;SET PARITY VECTOR TO 10\$
;FORCE GPO PARITY BITS TO 0 ON 2ND PASS
;THIS IS A SYNC POINT FOR SCOPING
;THE LOW BYTE HAS ITS P.B. OFF
;DATA FETCH SHOULD CAUSE PARITY ABORT
;SAVE MAINT REG IN CASE NO TRAP
;CLEAR MAINT REG IN CASE NO TRAP
;DUMMY INST WITH P.B.'S OFF
;FORCE GROUP 0 LOW BYTE PARITY BIT
;TO 0 AND MAIN MEMORY EVEN WORD HIGH

4370
4371
4372
4373 022666 012737 000030 177746
4374 022674 011401
4375 022676 005001
4376 022700 012737 023004 000114
4377 022706 010137 177750
4378 022712 000241
4379
4380 022714 011401
4381 022716 011503
4382 022720 105015
4383 022722 006701
4384 022724 012701 020020
4385

MOV #30,@#CONTRL
MOV (R4),R1
CLR R1
MOV #10\$,CACHVEC
MOV R1,@#MAINT
CLC
MOV (R4),R1
MOV (R5),R3
CLRB (R5)
SXT R1
MOV #020020,R1


```
4386
4387 022730 005227
4388 022732 177777
4389 022734 001764
4390 022736 023737 001200 001274
4391 022744 001402
4392 022746 010302
4393 022750 104051
4394 022752 012737 177777 177744
4395 022760 012737 005764 000114
4396 022766 013737 001202 177746
4397 022774 012737 022566 001110
4398 023002 000420
4399
4400
4401 023004 013737 177740 001270
4402 023012 013737 177742 001272
4403 023020 013737 177744 001274
4404 023026 013737 177746 001276
4405 023034 012737 030060 001300
4406 023042 000002
4407
4408
4409
4410
4411
4412
4413
4414
4415
4416
4417
4418
4419
4420
4421
4422
4423
4424
4425
4426
4427 023044 000004
4428 023046 012737 023332 001324
4429
4430 023054 012737 177777 023220
4431 023062 032737 000010 177746
4432 023070 001120
4433 023072 013737 177746 001202
4434 023100 012737 023604 001200
4435 023106 005037 001274
4436 023112 012737 177777 177744
4437 023120 013737 001240 172354
4438 023126 005077 156114
4439 023132 012777 040000 156104
4440 023140 012705 177750
4441 023144 012704 140000
```

```
21$: INC (PC)+
      .WORD -1
      BEQ 1$
      CMP $TMP4,PPARER
      BEQ 2$
      MOV R3,R2
      ERROR 51
2$: MOV #-1,MEMERR
      MOV #MEMER,CACHVEC
      MOV $TMP5,CONTRL
      MOV #20$, $LPERR
      BR TST44
      ;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
      ;FAILURE IN PARITY REPORTING
      ;CLEAR MEMORY ERROR REGISTER
      ;RESTORE PARITY SERVICE ROUTINE
      ;RESTORE CACHE TO FORMER CONDITION
      ;SET LOOP POINTER TO START OF TEST
      ;;TEST OVER GO TO NEXT TEST
```

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

```
*****
*TEST 44 PARITY REPORTING THRU THE MAP, CACHE DATA GROUP 1
*
* THIS TEST FORCES A PARITY ERROR IN CACHE GROUP 1 AT PHYSICAL
* ADDRESS 00040000 BY SETTING BIT 15 IN THE WORD. THE TEST
* THEN FORCES GROUP 1 LOW BYTE PARITY BIT TO ZERO AND EVEN
* WORD HIGH BYTE PARITY BIT TO ONE BEFORE READING ADDRESS 00040000
* THRU THE UNIBUS MAP. THEN IT CHECKS TO SEE THAT THE PARITY
* ABORT OCCURRED AND THAT THE CONDITION WAS CORRECT BEFORE
* GOING TO THE NEXT TEST.
*
* ERRORS ARE REPORTED INLINE IN THE TEST CODE, AND THE PARITY
* VECTOR WILL TRAP TO LABEL 10$ AT THE END OF THIS TEST.
*
* THIS TEST IS BEING RUN WITH ALL MEMORY REFERENCES GOING THRU
* THE UNIBUS MAP.
*****
```

```
TST44: SCOPE
      MOV #TST45,NXTTST ;SET UP ESCAPE POINTER IN CASE OF
      ;RANDOM PARITY ABORTS
20$: MOV #-1,21$ ;RESTORE NEG ONE FOR NEXT ITERATION
      BIT #BIT3,CONTRL ;SEE IF GROUP 1 IS DISABLED
      BNE TST45 ;BRANCH TO NEXT TEST IF GROUP 1 DISABLED
      MOV CONTRL,$TMP5 ;SAVE CONDITION OF CACHE
      MOV #23604,$TMP4 ;EXPECTED PARITY CONDITION GROUP 1
      CLR PPARER ;CLEAR MEM ERROR REG STORAGE
      MOV #-1,@#MEMERR ;CLEAR MEMORY ERROR REGISTER
      MOV LOWEST,KIPAR6 ;PUT UNIBUS ADDR OF MAP REG IN PAR6
      CLR @LREGU ;CLEAR UPPER BITS ON MAP REG
      MOV #40000,@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
```

```
4442 023150 012714 100000      MOV      #BIT15,(R4)      ;MAKE SURE I GET SOFT AND HARD ERRORS
4443                                     ;BY LOADING 40000 WITH BIT 15, THIS
4444                                     ;CAUSES THE HIGH BYTE P.B. TO BE ON
4445                                     ;AND THE LOW BYTE P.B. TO BE OFF
4446 023154 012737 000044 177746      MOV      #44,@#CONTRL      ;FORCE SELECTION OF GROUP 0 ON READ
4447 023162 011401      MOV      (R4),R1      ;PUT (40000) & (40002) IN GROUP 1
4448 023164 005001      CLR      R1      ;CLEAR R1 FOR MOMENT
4449 023166 012737 023272 000114      MOV      #10$,CACHVEC      ;SET PARITY VECTOR TO 10$
4450 023174 010137 177750 1$:      MOV      R1,@#MAINT      ;FORCE GP1 PARITY BITS TO 0 ON 2ND PASS
4451 023200 000241      CLC      ;THIS IS A SYNC POINT FOR SCOPING
4452                                     ;THE LOW BYTE HAS ITS P.B. OFF
4453 023202 011401      MOV      (R4),R1      ;DATA FETCH SHOULD CAUSE PARITY ABORT
4454 023204 011503      MOV      (R5),R3      ;SAVE MAINT REG IN CASE NO TRAP
4455 023206 105015      CLRB     (R5)      ;CLEAR MAINT REG IN CASE NO TRAP
4456 023210 006701      SXT      R1      ;DUMMY INST WITH P.B.'S OFF
4457 023212 012701 020100      MOV      #020100,R1      ;FORCE GROUP 1 LOW BYTE PARITY BIT
4458                                     ;TO 0 AND MAIN MEMORY EVEN WORD HIGH
4459                                     ;BYTE PARITY BIT TO 1
4460 023216 005227      INC      (PC)+      ;INCREMENT NEXT WORD TO 0
4461 023220 177777 21$:      .WORD     -1      ;
4462 023222 001764      BEQ      1$      ;BRANCH ON FIRST PASS ONLY
4463 023224 023737 001200 001274      CMP      $TMP4,PPARER      ;SEE IF ERROR CONDITION MATCHES EXPECTED
4464 023232 001402      BEQ      2$      ;BRANCH IF ERROR WAS EXPECTED ONE
4465 023234 010302      MOV      R3,R2      ;R2 HOLDS MAINT REG FOR ERROR TYPE OUT
4466 023236 104051      ERROR     51      ;FAILURE IN PARITY REPORTING
4467 023240 012737 177777 177744 2$:      MOV      #-1,MEMERR      ;CLEAR MEMORY ERROR REGISTER
4468 023246 012737 005764 000114      MOV      #MEMER,CACHVEC      ;RESTORE PARITY SERVICE ROUTINE
4469 023254 013737 001202 177746      MOV      $TMP5,CONTRL      ;RESTORE CACHE TO FORMER CONDITION
4470 023262 012737 023054 001110      MOV      #20$, $LPERR      ;SET LOOP POINTER TO START OF TEST
4471 023270 000420      BR       TST45      ;:TEST OVER GO TO NEXT TEST
4472
4473
4474 023272 013737 177740 001270 10$:      MOV      @#LOADRS,PLOADR      ;SAVE LOWER CACHE ADDR REG
4475 023300 013737 177742 001272      MOV      @#HIADRS,PHIADR      ;SAVE HIGH BITS OF FAILING ADDR
4476 023306 013737 177744 001274      MOV      @#MEMERR,PPARER      ;SAVE MEMORY ERROR REGISTER
4477 023314 013737 177746 001276      MOV      @#CONTRL,PCONTR      ;SAVE CONTROL REGISTER FOR TYPE OUT
4478 023322 012737 030300 001300      MOV      #30300,PMAINT      ;SAVE DATA IN MAINTENANCE REGISTER
4479 023330 000002      RTI      ;RETURN TO TEST AND CHECK PARITY
4480
4481
4482
4483
4484
4485
4486
4487 023332 000004      TST45:      SCOPE
4488 023334 012737 023466 001324      MOV      #TST46,NXTTST
4489 023342 105737 001331      TSTB     KB11EM      ;IS THIS A KB11-EM?
4490 023346 001003      BNE      10$      ;BP IF YES
4491 023350 105737 001332      TSTB     KB11CM      ;IS THIS A MODIFIED PROCESSOR?
4492 023354 001444      BEQ      5$
4493 023356 012700 170202 10$:      MOV      #170202,R0
4494 023362 005010 4$:      CLR      (R0)
4495 023364 032710 100000      BIT      #BYP,(R0)
4496 023370 001405      BEQ      1$
4497 023372 010037 001154      MOV      R0,$REG0
```


4498	023376	011037	001156
4499	023402	104053	
4500			
4501	023404	052710	100000
4502	023410	032710	100000
4503	023414	001005	
4504	023416	010037	001154
4505	023422	011037	001156
4506	023426	104054	
4507			
4508	023430	042710	100000
4509	023434	032710	100000
4510	023440	001405	
4511	023442	010037	001154
4512	023446	011037	001156
4513	023452	104053	

	MOV	(R0), \$REG1
	ERROR	53
1\$:	BIS	#BYP, (R0)
	BIT	#BYP, (R0)
	BNE	2\$
	MOV	R0, \$REG0
	MOV	(R0), \$REG1
	ERROR	54
2\$:	BIC	#BYP, (R0)
	BIT	#BYP, (R0)
	BEQ	3\$
	MOV	R0, \$REG0
	MOV	(R0), \$REG1
	ERROR	53

PDP-11/70-74MP UNIBUS MAP DIAGNOSTIC
CEKBFC.P11 03-MAY-79 13:19

J 8
MACY11 30A(1052) 03-MAY-79 13:21 PAGE 89
T45 CHECK BYP BITS IN UBM

SEQ 0100

4514
4515 023454 062700 000004
4516 023460 020027 170366
4517 023464 001336
4518 023466

3\$: ADD #4,R0
CMP R0,#170366
BNE 4\$
5\$:

4519
4520
4521
4522
4523
4524
4525
4526 023466 000004

: *TEST 46 CHECK FOR PRESENCE OF MKA11 REGS. ON KB11-EM OR KB11CM
: * THIS TEST IS EXECUTED ONLY ON KB11-EM OR 11/74 (KB11CM)
: *
: *****
TST46: SCOPE


```
4527 023470 012737 023626 001324      MOV    #TST47,NXTTST
4528      172100      MKCSR=172100
4529
4530      177744      MSER= 177744
4531 023476 105737 001331      TSTB   KB11EM      ;KB11-EM?
4532 023502 001003      BNE     1$      ;BR IF YES
4533 023504 105737 001332      TSTB   KB11CM      ;MODIFIED CPU?
4534 023510 001446      BEQ     99$      ;NO, SKIP THIS TEST
4535 023512 012702 172100      1$: MOV    #MKCSR, R2      ;POINT TO CSR REGISTERS.
4536 023516 013746 000004      MOV    @#4, -(SP)      ;SAVE OLD CPU ERROR VEC
4537 023522 012737 023546 000004      MOV    #66$, @#4      ;REPLACE WITH TEST ROUTINE
4538
4539 023530 005712      2$: TST     (R2)      ;TRY TO ACCESS CSR
4540 023532 062702 000002      ADD     #2,R2      ;INCREMENT R2 BY 2
4541 023536 020227 172140      4$: CMP    R2,#172140      ;ALL REGISTERS TESTED?
4542 023542 103772      BLO     2$      ;NOPE, TEST SOMEMORE
4543 023544 000426      BR      98$      ;TEST IS OVER.
4544      ;:****      TRAP HANDLER      ***
4545 023546 012705 001000      66$: MOV    #1000,R5      ;SET COUNTER AND...
4546 023552 077501      SOB     R5,,      ;WAIT A WHILE FOR THE GATES TO SETTLE
4547 023554 005737 177744      TST     @#MSER      ;ANY BITS SET IN MEM. ERR. REG?
4548 023560 001004      BNE     68$      ;YES, ERROR
4549 023562 032737 000020 177766      BIT     #BIT4,@#CPUERR ;UNIBUS ERROR BIT SET?
4550 023570 001011      BNE     77$      ;YES,OK
4551 023572 010237 001160      68$: MOV    R2,$REG2      ;PRINT OFFENDING CSR
4552 023576 013737 177744 001172      MOV    @#MSER,$TMP1      ;MSER
4553 023604 013737 177766 001174      MOV    @#CPUERR,$TMP2    ;CPUERR
4554 023612 104055      ERROR   55      ;REPORT ERROR
4555 023614 005037 177766      77$: CLR    @#CPUERR      ;CLEAR THE ERROR REGISTER
4556 023620 000002      RTI
4557      ;:*****
4558
4559 023622 012637 000004      98$: MOV    (SP)+, @#4      ;RESTORE CPU ERROR VECTOR
4560
4561 023626      99$:
4562      ;:*****
4563      ;*TEST 47      CHECK CACHE BYPASS ON UNIBUS MAP PAGE
4564      ;THIS TEST CHECKS CACHE BYPASS ON UNIBUS-MAP-PAGE
4565      ;THIS TEST IS EXECUTED ONLY ON KB11-EM OR 11/74      (KB11CM)
4566      ;THIS TEST USES MEMORY MANAGMENT
4567      ;:*****
4568 023626 000004      TST47: SCOPE
4569 023630 012737 000005 001204      MOV    #5,$TIMES      ;:DO 5 ITERATIONS
4570
4571 023636 013746 177776      MOV    @#PS,-(SP)
4572 023642 042716 000020      BIC    #20,(SP)      ;CLEAR T BIT IF SET
4573 023646 012746 023654      MOV    #1$,-(SP)
4574 023652 000002      RTI
4575 023654      1$:
4576
4577 023654 012737 024346 001324      MOV    #SEOP,NXTTST
4578 023662 105737 001331      TSTB   KB11EM      ;IS THIS A KB11-EM?
4579 023666 001005      BNE     CBMP      ;BR IF YES
4580 023670 105737 001332      TSTB   KB11CM
4581 023674 001002      BNE     CBMP
4582 023676 000137 024346      JMP     SEOP
```

```
4583 023702 012737 170000 172354 CBMP: MOV #170000,@#KIPAR6 ;MAP TO UNIBUS MAPPING BOX
4584 023710 012737 010406 172314 MOV #10406,@#KIPDR6 ;THROUGH PAGE 6
4585
4586 023716 012705 170200 MOV #MAPL00,R5 ;STARTING ADDRESS OF UBM'R'S
4587
4588 023722 005037 177746 CBMPA: CLR @#CONTRL
4589 023726 032737 010000 177746 1$: BIT #VCIP,@#CONTRL
4590 023734 001374 BNE 1$
4591
4592 023736 012700 023702 MOV #CBMP,R0 ;MAKE TEST CODE HIT IN GROUP 0
4593 023742 012702 001000 MOV #1000,R2
4594 023746 012737 000054 177746 2$: MOV #S1MOM1,@#CONTRL
4595 023754 005760 002000 TST 2000(R0)
4596 023760 012737 000034 177746 MOV #SOMOM1,@#CONTRL
4597 023766 005720 TST (R0)+
4598 023770 077212 SOB R2,2$ ;DONE?
4599
4600 023772 012700 040050 MOV #TSTDAT,R0 ;MAKE TEST DATA HIT IN GROUP 1
4601 023776 012702 001000 MOV #1000,R2
4602 024002 012737 000040 177746 9$: MOV #S1,@#CONTRL
4603 024010 005720 TST (R0)+
4604 024012 077202 SOB R2,9$
4605
4606 024014 012737 177600 172356 MOV #177600,@#KIPAR7 ;MAP I/O PAGE THROUGH PAGE 7
4607 024022 012737 077406 172316 MOV #77406,@#KIPDR7
4608 024030 005037 172340 CLR @#KIPAR0 ;MAP 0-4 K VIRTUAL INTO
4609 024034 012737 077406 172300 MOV #77406,@#KIPDR0 ;0-4 K PHYSICAL ADDRESS FORCE
4610 024042 012737 000200 172342 MOV #200,@#KIPAR1 ;MAP 4-8K VIRTUAL INTO 4-8K
4611 024050 012737 077406 172302 MOV #77406,@#KIPDR1 ;PHYSICAL ADDRESS SPACE
4612 024056 012737 000400 172344 MOV #400,@#KIPAR2
4613 024064 012737 077406 172304 MOV #77406,@#KIPDR2
4614
4615 024072 012700 040050 MOV #TSTDAT,R0 ;SET UP UNIBUS MAPPING
4616 024076 042700 000077 BIC #77,R0 ;REGISTER. NOTE THIS REGISTER
4617 024102 010025 MOV R0,(R5)+ ;WILL BE USED FOR CACHE-
4618 024104 005015 CLR (R5) ;BYPASS ON UNIBUS MAP PAGE.
4619
4620 024106 012700 040050 MOV #TSTDAT,R0 ;FORM THE VIRTUAL ADDRESS FOR
4621 024112 042700 177700 BIC #177700,R0 ;THAT DATA. NOTE TEST DATA
4622 024116 052700 140000 BIS #140000,R0 ;IS MAPPED THROUGH PAR6
4623 024122 010001 MOV R0,R1 ;R0 CONTAINS THE VA
4624 024124 012702 001000 MOV #1000,R2
4625
4626 024130 012737 000060 172516 MOV #60,@#MMR3 ;ENABLE 22-BIT MAPPING
4627 024136 012737 000001 177572 MOV #1,@#MMR0 ;TURN ON KT AND UNIBUS MAP
4628
4629 024144 005710 3$: TST (R0) ;REFERENCE TEST-DATA,CHECK IT
4630 024146 032737 000010 177752 BIT #10,@#HITMIS ;IS A HIT
4631 024154 001011 BNE 4$
4632 024156 013737 177746 001154 MOV @#CONTRL,$REGO
4633 024164 012737 000001 001156 MOV #1,$REG1
4634 024172 010037 001160 MOV R0,$REG2
4635 024176 104056 ERROR 56
4636
4637
4638 ;GROUP NO, BEING TESTED
;TEST-DATA ADDRESS
;TEST DATA REFERENCE NOT
;A HIT, AS EXPECTED. IT WAS
;MADE BIT IN GROUP 1,
;PREVIOUSLY. THEN IT WAS
```


4639							:REFERENCE THROUGH THE UNIBUS
4640							:MAP-
4641							
4642	024200	062700	000002	4\$:	ADD	#2,R0	
4643	024204	077221			SOB	R2,3\$	:DONE?
4644							
4645	024206	052715	100000		BIS	#BYP,(R5)	:GET BYPASS IN (HI) UNIBUS
4646							:MAP REGISTER BEING CHECKED
4647	024212	012702	001000		MOV	#1000,R2	
4648	024216	010100			MOV	R1,R0	:R0 CONTAINS THE VA OF TEST-DATA
4649	024220	005711		5\$:	TST	(R1)	:REFERENCE TEST DATA THROUGH
4650							:THE UNIBUS MAP (WITH CACHE
4651							:BYPASS SET) AND CHECK
4652							:IF THE REFERENCE WAS A MISS
4653							
4654	024222	062701	000002	6\$:	ADD	#2,R1	
4655	024226	077204			SOB	R2,5\$	:DONE?
4656							
4657	024230	005037	177572		CLR	@MMR0	:TURN OFF MEM MNGMNT
4658	024234	005037	172516		CLR	@MMR3	
4659	024240	042715	100000		BIC	#BYP,(R5)	:CLEAR CACHE BYPASS IN UBM R
4660	024244	012700	040050		MOV	#TSTDAT,R0	
4661	024250	012702	000400		MOV	#400,R2	
4662	024254	005710		7\$:	TST	(R0)	:REFERENCE TEST-DATA AGAIN
4663							:AND CHECK THAT IT WAS
4664							:INVALIDATED PREVIOUSLY
4665	024256	032737	000010		BIT	#10,@#HITMIS	:MISS?
4666	024264	001413			BEQ	8\$	:YES
4667	024266	013737	177746		MOV	@CONTRL,\$REGO	
4668	024274	010537	001156		MOV	R5,\$REG1	:ADDRESS OF UB MAP REGISTER THAT
4669	024300	162737	000002		SUB	#2,\$REG1	:IS BEING TESTED
4670	024306	010037	001160		MOV	R0,\$REG2	:TEST-DATA ADDRESS
4671	024312	104057			ERROR	57	:TEST DATA REFERENCE WAS NOT
4672							:A MISS. TEST-DATA WAS
4673							:PREVIOUSLY REFERENCED IN
4674							:BYPASS (ON UBM R) MODE,HENCE
4675							:IT SHOULD HAVE BEEN
4676							:INVALIDATED. BUT, SUBSEQUENT
4677							:REFERENCE WAS NOT A MISS,
4678							:AS EXPECTED
4679							
4680	024314	062700	000004	8\$:	ADD	#4,R0	
4681	024320	077223			SOB	R2,7\$	:DONE?
4682	024322	020527	170366		CMP	R5,#MAPH35	:DONE CHECKING ALL 31 UBM R'S?
4683	024326	001407			BEQ	10\$	
4684							
4685	024330	062705	000002		ADD	#2,R5	:ADDRESS OF NEXT UBM R
4686	024334	062737	000200		ADD	#200,@#KIPAR6	:MAP PAR 6 TO RESET THE
4687	024342	000137	023722		JMP	CBMPA	:NEXT UBM R (TO BE TESTED)
4688	024346			10\$:			
4689							
4690							
4691							
4692							
4693							
4694							

```

4695      ;*INDICATE END-OF-PROGRAM AFTER 1 PASSES THRU THE PROGRAM
4696      ;*TYPE 'END PASS #XXXXX TOTAL NUMBER OF ERRORS SINCE LAST REPORT YYYYYY'
4697      ;*WHERE XXXXX AND YYYYY ARE DECIMAL NUMBERS
4698      ;*IF SW12=1 INHIBIT TRACE TRAP
4699      ;*IF THERES A MONITOR GO TO IT
4700      ;*IF THERE ISN'T JUMP TO LOOP
4701
4702      $EOP:
4703      SCOPE
4704      CLR      MMRO
4705      CLR      MMR3
4706      TBTR
4707      CLR      $TSTNM
4708      CLR      $TIMES
4709      INC      $PASS
4710      BIC      #100000,$PASS
4711      DEC      (PC)+
4712      $EOPCT: .WORD 1
4713      BGT      $DOAGN
4714      MOV      (PC)+,@(PC)+
4715      $ENDCT: .WORD 1
4716      $EOPCT
4717      TYPE     ,65$
4718      BR       64$
4719      ;:65$: .ASCIZ <12><15>/END PASS #/
4720      64$:
4721      MOV      $PASS,-(SP)
4722
4723      TYPDS
4724      TYPE     ,67$
4725      BR       66$
4726      ;:67$: .ASCIZ / TOTAL ERRORS SINCE LAST REPORT /
4727      66$:
4728      MOV      $ERTTL,-(SP)
4729
4730      TYPDS
4731      TYPE     , $CRLF
4732      CLR      $ERTTL
4733      $GET42: MOV      @#42,R0
4734      BEQ      $DOAGN
4735      CLR      -(SP)
4736      MOV      # $CLR.T,-(SP)
4737      BR       $RTRN
4738
4739      $CLR.T:
4740      MOV      @#42,R0
4741      BEQ      $DOAGN
4742      RESET
4743      $ENDAD: JSR      PC,(R0)
4744      NOP
4745      NOP
4746      NOP
4747      $DOAGN:
4748      MOV      @#PS,-(SP)
4749      BIC      #20,(SP)
4750      BIT      #BIT12,@#SWR

```

```

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

```



```
4751 024614 001005          BNE      1$          ;;BR IF NO
4752 024616 005137 024642    COM      $TBIT      ;;IS IT TIME FOR TRACE TRAP
4753 024622 100402          BMI      1$          ;;BR IF NO
4754 024624 052716 000020    BIS      #20,(SP)    ;;SET TRACE TRAP
4755 024630 012746 024636    1$:      MOV      #$LOOP,-(SP) ;;JUMP TO START OF TEST
4756 024634 000002          $RTN:   RTI          ;;RETURN--THIS IS CHANGED TO
4757                                     ;;AN 'RTT' IF 'RTT' IS A LEGAL
4758                                     ;;INSTRUCTION
4759 024636          $LOOP:   JMP      @#LOOP      ;;RETURN
4760 024636 000137 010676    $TBIT:   .WORD      0      ;;'T' BIT STATE INDICATOR
4761 024642 000000          $ENULL:  .BYTE      -1,-1,0  ;;NULL CHARACTER STRING
4762 024644          377      377      000
4763 024650          024650
4764
4765          :MESSAGES
4766 024650 041600 052520 052440 MSG1:   .ASCIZ<CRLF>    'CPU UNDER TEST FOUND TO BE A ''
4767 024656 042116 051105 052040
4768 024664 051505 020124 047506
4769 024672 047125 020104 047524
4770 024700 041040 020105 020101
4771 024706          000
4772 024707          113 030502 026461 MSG2:   .ASCIZ  'KB11-EM'<CRLF>
4773 024714 046505 000200
4774 024720 041113 030461 041055 MSG3:   .ASCIZ  'KB11-B/C'<CRLF>
4775 024726 041457 000200
4776 024732 041113 030461 041455 MSG4:   .ASCIZ  'KB11-CM          '<CRLF>
4777 024740 020115 020040 020040
4778 024746 020040 020040 020040
4779 024754 020040 020040 100040
4780 024762          000
4781 024763          113 030502 026461 MSG5:   .ASCIZ  'KB11-E'<CRLF>
4782 024770 100105          000
4783
4784
4785          .SBTTL  ERROR MESSAGES AND DATA TABLES
4786 024773          116 052117 052040 EM1:   .ASCIZ  ?NOT THE CORRECT TRAP CONDITION THRU ERRVEC (#004)?
4787 025000 042510 041440 051117
4788 025006 042522 052103 052040
4789 025014 040522 020120 047503
4790 025022 042116 052111 047511
4791 025030 020116 044124 052522
4792 025036 042440 051122 042526
4793 025044 020103 021450 030060
4794 025052 024464          000
4795 025055          125 042516 050130 EM2:   .ASCIZ  ?UNEXPECTED CPU TRAP THRU ERRVEC (#004)?
4796 025062 041505 042524 020104
4797 025070 050103 020125 051124
4798 025076 050101 052040 051110
4799 025104 020125 051105 053122
4800 025112 041505 024040 030043
4801 025120 032060 000051
4802 025124 047125 054105 042520 EM3:   .ASCIZ  ?UNEXPECTED CACHE PARITY ERROR THRU CACHVEC, WILL RETRY TEST ONCE?
4803 025132 052103 042105 041440
4804 025140 041501 042510 050040
4805 025146 051101 052111 020131
4806 025154 051105 047522 020122
```

4807	025162	044124	052522	041440
4808	025170	041501	053110	041505
4809	025176	020054	044527	046114
4810	025204	051040	052105	054522
4811	025212	052040	051505	020124
4812	025220	047117	042503	000
4813	025225	125	042516	050130
4814	025232	041505	042524	020104
4815	025240	040515	047111	046440
4816	025246	046505	051117	020131
4817	025254	040520	044522	054524
4818	025262	042440	051122	051117
4819	025270	052040	051110	020125
4820	025276	040503	044103	042526
4821	025304	026103	053440	046111
4822	025312	020114	042522	051124
4823	025320	020131	042524	052123
4824	025326	047440	041516	000105
4825	025334	047125	054105	042520
4826	025342	052103	042105	046440
4827	025350	046505	051117	020131
4828	025356	040515	040516	042507
4829	025364	042515	052116	052040
4830	025372	040522	026120	046440
4831	025400	046505	051117	020131
4832	025406	040515	040516	042507
4833	025414	042515	052116	051440
4834	025422	040524	052524	020123
4835	025430	042522	044507	052123
4836	025436	051105	000123	
4837	025442	052523	046515	051101
4838	025450	020131	043117	046440
4839	025456	050101	051040	043505
4840	025464	051511	042524	051522
4841	025472	052040	040510	020124
4842	025500	044524	042515	020104
4843	025506	052517	020124	047117
4844	025514	051040	040505	000104
4845	025522	052523	046515	051101
4846	025530	020131	043117	041440
4847	025536	041501	042510	051040
4848	025544	043505	051511	042524
4849	025552	051522	052040	040510
4850	025560	020124	044524	042515
4851	025566	020104	052517	020124
4852	025574	047117	051040	040505
4853	025602	000104		
4854	025604	052523	046515	051101
4855	025612	020131	043117	046440
4856	025620	050101	051040	043505
4857	025626	051511	042524	051522
4858	025634	047040	052117	044040
4859	025642	046117	044504	043516
4860	025650	055040	051105	020117
4861	025656	047111	046040	053517
4862	025664	030440	020066	044502

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?

4863	025672	051524	000
4864	025675	123	046525
4865	025702	054522	047440
4866	025710	040515	020120
4867	025716	044507	052123
4868	025724	020123	047516
4869	025732	047510	042114
4870	025740	020107	042532
4871	025746	044440	020116
4872	025754	042520	020122
4873	025762	044502	051524
4874	025767	120	051517
4875	025774	046102	020105
4876	026002	047522	020122
4877	026010	046440	050101
4878	026016	043505	051511
4879	026024	020122	040504
4880	026032	050040	052101
4881	026040	046450	050101
4882	026046	043505	030040
4883	026054	000	
4884	026055	116	053517
4885	026062	047522	040502
4886	026070	020105	051105
4887	026076	020122	047111
4888	026104	050101	051040
4889	026112	051511	042524
4890	026120	040504	040524
4891	026126	052101	020110
4892	026134	050101	051040
4893	026142	031040	024460
4894	026147	123	046525
4895	026154	054522	047440
4896	026162	052504	046101
4897	026170	042104	042522
4898	026176	047111	020107
4899	026204	047522	051522
4900	026212	020116	047514
4901	026220	047111	020107
4902	026226	020120	042522
4903	026234	052123	051105
4904	026242	052523	046515
4905	026250	020131	043117
4906	026256	052517	052116
4907	026264	052101	042524
4908	026272	043040	044501
4909	026300	042522	020123
4910	026306	046040	053517
4911	026314	030440	020066
4912	026322	051524	047440
4913	026330	040515	020120
4914	026336	044507	052123
4915	026344	000123	
4916	026346	052523	046515
4917	026354	020131	043117
4918	026362	052517	052116

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

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

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

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

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

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

4919	026370	052101	042524	047122
4920	026376	043040	044501	052514
4921	026404	042522	020123	047111
4922	026412	052440	050120	051105
4923	026420	033040	041040	052111
4924	026426	020123	043117	046440
4925	026434	050101	051040	043505
4926	026442	051511	042524	051522
4927	026450	000		
4928	026451	103	052517	042114
4929	026456	047040	052117	041440
4930	026464	042514	051101	041440
4931	026472	041501	042510	041440
4932	026500	047117	051124	046117
4933	026506	051040	043505	051511
4934	026514	042524	100122	
4935	026520	047520	051523	041111
4936	026526	042514	042440	051122
4937	026534	051117	044440	020116
4938	026542	040503	044103	020105
4939	026550	042522	044507	052123
4940	026556	051105	042040	052101
4941	026564	020101	040520	044124
4942	026572	000		
4943	026573	103	052517	042114
4944	026600	047040	052117	041440
4945	026606	042514	051101	041440
4946	026614	041501	042510	046440
4947	026622	044501	052116	047105
4948	026630	047105	042503	051040
4949	026636	043505	051511	042524
4950	026644	100122		
4951	026646	047520	051523	041111
4952	026654	042514	042440	051122
4953	026662	051117	044440	020116
4954	026670	040503	044103	020105
4955	026676	042522	044507	052123
4956	026704	051105	042040	052101
4957	026712	020101	040520	044124
4958	026720	000		
4959	026721	103	052517	042114
4960	026726	047040	052117	051040
4961	026734	040505	020104	033461
4962	026742	033467	030064	043040
4963	026750	047522	020115	040503
4964	026756	044103	020105	047514
4965	026764	040440	042104	042522
4966	026772	051523	051040	043505
4967	027000	024040	047514	042101
4968	027006	051522	100051	
4969	027012	047520	051523	041111
4970	027020	042514	042440	051122
4971	027026	051117	044440	020116
4972	027034	040503	044103	020105
4973	027042	042522	044507	052123
4974	027050	051105	042040	052101

EM17: .ASCII ?COULD NOT CLEAR CACHE CONTROL REGISTER?<CRLF>

.ASCIIZ ?POSSIBLE ERROR IN CACHE REGISTER DATA PATH?

EM20: .ASCII ?COULD NOT CLEAR CACHE MAINTENENCE REGISTER?<CRLF>

.ASCIIZ ?POSSIBLE ERROR IN CACHE REGISTER DATA PATH?

EM21: .ASCII ?COULD NOT READ 177740 FROM CACHE LO ADDRESS REG (LOADRS)?<CRLF>

.ASCIIZ ?POSSIBLE ERROR IN CACHE REGISTER DATA PATH?

4975	027056	020101	040520	044124
4976	027064	000		
4977	027065	103	052517	042114
4978	027072	047040	052117	051040
4979	027100	040505	020104	030060
4980	027106	030060	031460	043040
4981	027114	047522	020115	040503
4982	027122	044103	020105	044510
4983	027130	040440	042104	042522
4984	027136	051523	051040	043505
4985	027144	024040	044510	042101
4986	027152	051522	100051	
4987	027156	047520	051523	041111
4988	027164	042514	042440	051122
4989	027172	051117	044440	020116
4990	027200	040503	044103	020105
4991	027206	042522	044507	052123
4992	027214	051105	042040	052101
4993	027222	020101	040520	044124
4994	027230	000		
4995	027231	123	046525	040515
4996	027236	054522	047440	020106
4997	027244	047503	047125	020124
4998	027252	040520	052124	051105
4999	027260	020116	040506	046111
5000	027266	051125	051505	044440
5001	027274	020116	040503	044103
5002	027302	020105	047503	052116
5003	027310	047522	020114	042522
5004	027316	044507	052123	051105
5005	027324	000		
5006	027325	123	046525	040515
5007	027332	054522	047440	020106
5008	027340	047503	047125	020124
5009	027346	040520	052124	051105
5010	027354	020116	040506	046111
5011	027362	051125	051505	044440
5012	027370	020116	040503	044103
5013	027376	020105	040515	047111
5014	027404	042524	042516	041516
5015	027412	020105	042522	044507
5016	027420	052123	051105	000
5017	027425	122	043105	051105
5018	027432	047105	042503	020104
5019	027440	040515	020120	042522
5020	027446	044507	052123	051105
5021	027454	030040	053440	052111
5022	027462	020110	042101	051104
5023	027470	051505	020123	047117
5024	027476	020105	044502	020124
5025	027504	044504	043106	051105
5026	027512	047105	020124	044124
5027	027520	047101	033440	030067
5028	027526	030062	000060	
5029	027532	042522	042506	042522
5030	027540	041516	042105	041440

EM22: .ASCII ?COULD NOT READ 000003 FROM CACHE HI ADDRESS REG (HIADRS)?<CRLF>

.ASCIIZ ?POSSIBLE ERROR IN CACHE REGISTER DATA PATH?

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

EM24: .ASCIIZ ?SUMMARY OF COUNT PATTERN FAILURES IN CACHE MAINTENENCE REGISTER?

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

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

5031	027546	041501	042510	046040
5032	027554	053517	040440	042104
5033	027562	042522	051523	051040
5034	027570	043505	051511	042524
5035	027576	020122	044527	044124
5036	027604	040440	042104	042522
5037	027612	051523	047440	042516
5038	027620	041040	052111	200
5039	027625	104	043111	042506
5040	027632	042522	052116	052040
5041	027640	040510	020116	033467
5042	027646	033467	030064	000
5043	027653	103	047101	052047
5044	027660	043440	052105	052040
5045	027666	020117	040515	047111
5046	027674	046440	046505	051117
5047	027702	020131	051106	046517
5048	027710	052440	044516	052502
5049	027716	020123	044527	044124
5050	027724	052040	042510	046440
5051	027732	050101	047440	043106
5052	027740	200		
5053	027741	123	020117	023511
5054	027746	046114	045040	046525
5055	027754	020120	047524	052040
5056	027762	042510	051440	055111
5057	027770	020105	052512	050115
5058	027776	051105	052040	051505
5059	030004	020124	047506	020122
5060	030012	042526	044522	044506
5061	030020	040503	044524	047117
5062	030026	000		
5063	030027	123	046525	040515
5064	030034	054522	047440	020106
5065	030042	047503	047125	020124
5066	030050	040520	052124	051105
5067	030056	020116	040506	046111
5068	030064	051125	051505	047440
5069	030072	020116	044124	020105
5070	030100	047125	041111	051525
5071	030106	042040	052101	020101
5072	030114	040520	044124	000
5073	030121	125	044516	052502
5074	030126	020123	040515	020120
5075	030134	051511	051040	046105
5076	030142	041517	052101	047111
5077	030150	020107	044127	047105
5078	030156	047040	052117	042440
5079	030164	040516	046102	042105
5080	030172	000		
5081	030173	103	047101	047516
5082	030200	020124	051525	020105
5083	030206	047101	020131	043117
5084	030214	052040	042510	046440
5085	030222	050101	051040	043505
5086	030230	051511	042524	051522

. ASCIIZ ?DIFFERENT THAN 777740?

EM30: . ASCII ?CAN'T GET TO MAIN MEMORY FROM UNIBUS WITH THE MAP OFF?<CRLF>

. ASCIIZ ?SO I'LL JUMP TO THE SIZE JUMPER TEST FOR VERIFICATION?

EM31: . ASCIIZ ?SUMMARY OF COUNT PATTERN FAILURES ON THE UNIBUS DATA PATH?

EM32: . ASCIIZ ?UNIBUS MAP IS RELOCATING WHEN NOT ENABLED?

EM33: . ASCII ?CANNOT USE ANY OF THE MAP REGISTERS, OR PHYSICAL?<CRLF>

5087	030236	020054	051117	050040
5088	030244	054510	044523	040503
5089	030252	100114		
5090	030254	042101	051104	051505
5091	030262	020123	044502	030524
5092	030270	020064	051511	051440
5093	030276	052524	045503	046040
5094	030304	053517	020056	052515
5095	030312	052123	051040	051505
5096	030320	040524	052122	050040
5097	030326	047522	051107	046501
5098	030334	044440	020106	047516
5099	030342	046040	047517	000120
5100	030350	044124	020105	052516
5101	030356	041115	051105	047440
5102	030364	020106	040515	020120
5103	030372	042522	044507	052123
5104	030400	051105	020123	042522
5105	030406	047515	042526	020104
5106	030414	054502	045040	046525
5107	030422	042520	020122	042523
5108	030430	052124	047111	020107
5109	030436	047504	051505	200
5110	030443	116	052117	040440
5111	030450	051107	042505	053440
5112	030456	052111	020110	044124
5113	030464	020105	052516	041115
5114	030472	051105	043040	052517
5115	030500	042116	052040	020117
5116	030506	042502	046440	051511
5117	030514	044523	043516	000
5118	030521	124	042510	051440
5119	030526	055111	020105	052512
5120	030534	050115	051105	020123
5121	030542	047117	052040	042510
5122	030550	052440	044516	052502
5123	030556	020123	040515	020120
5124	030564	051101	020105	047516
5125	030572	020124	042523	020124
5126	030600	047111	200	
5127	030603	124	042510	051111
5128	030610	042040	043105	052501
5129	030616	052114	050040	051517
5130	030624	052111	047511	026116
5131	030632	053440	044510	044103
5132	030640	040440	046114	053517
5133	030646	020123	047125	041111
5134	030654	051525	040440	042104
5135	030662	042522	051523	051505
5136	030670	200		
5137	030671	060	030060	030060
5138	030676	020060	047524	033440
5139	030704	033465	033467	020066
5140	030712	047524	051040	043105
5141	030720	051105	047105	042503
5142	030726	046440	044501	020116

.ASCIZ ?ADDRESS BIT14 IS STUCK LOW. MUST RESTART PROGRAM IF NO LOOP?

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

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

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

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

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

5143	030734	042515	047515	054522
5144	030742	200		
5145	030743	124	042510	051111
5146	030750	041440	051125	042522
5147	030756	052116	051440	052105
5148	030764	044524	043516	040440
5149	030772	046114	053517	020123
5150	031000	047117	054514	000072
5151	031006	040515	020120	042522
5152	031014	044507	052123	051105
5153	031022	052440	042116	051105
5154	031030	052040	051505	020124
5155	031036	044504	020104	047516
5156	031044	020124	042522	050123
5157	031052	047117	020104	047111
5158	031060	042040	040525	020114
5159	031066	040515	050120	047111
5160	031074	020107	042524	052123
5161	031102	000		
5162	031103	123	046525	040515
5163	031110	054522	047440	020106
5164	031116	047125	041111	051525
5165	031124	040440	042104	042522
5166	031132	051523	042440	051122
5167	031140	051117	026123	053440
5168	031146	052111	020110	044124
5169	031154	020105	040515	020120
5170	031162	042522	047514	040503
5171	031170	044524	047117	042040
5172	031176	051511	041101	042514
5173	031204	000104		
5174	031206	040515	047111	046440
5175	031214	046505	051117	020131
5176	031222	044524	042515	052517
5177	031230	020124	053117	051105
5178	031236	052040	042510	052440
5179	031244	044516	052502	020123
5180	031252	044504	020104	047516
5181	031260	020124	041517	052503
5182	031266	020122	051120	050117
5183	031274	051105	054514	000
5184	031301	122	046105	041517
5185	031306	052101	047511	020116
5186	031314	044124	052522	052040
5187	031322	042510	046440	050101
5188	031330	053440	051501	047040
5189	031336	052117	041440	051117
5190	031344	042522	052103	020054
5191	031352	052506	046114	040440
5192	031360	042104	000	
5193	031363	122	046105	041517
5194	031370	052101	047511	020116
5195	031376	044124	052522	052040
5196	031404	042510	046440	050101
5197	031412	053440	051501	047040
5198	031420	052117	041440	051117

. ASCIIZ ?THEIR CURRENT SETTING ALLOWS ONLY:?

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

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

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

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

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

5199	031426	042522	052103	020054	
5200	031434	040503	051122	020131	
5201	031442	051120	050117	043501	
5202	031450	052101	047511	000116	
5203	031456	044124	020105	047524	EM43: .ASCIIZ ?THE TOP OF MEMORY IS DIFFERENT THAN THE SIZE JUMPERS?
5204	031464	020120	043117	046440	
5205	031472	046505	051117	020131	
5206	031500	051511	042040	043111	
5207	031506	042506	042522	052116	
5208	031514	052040	040510	020116	
5209	031522	044124	020105	044523	
5210	031530	042532	045040	046525	
5211	031536	042520	051522	000	
5212	031543	120	051101	052111	EM44: .ASCIIZ ?PARITY REPORTING THRU THE MAP IS NOT CORRECT?
5213	031550	020131	042522	047520	
5214	031556	052122	047111	020107	
5215	031564	044124	052522	052040	
5216	031572	042510	046440	050101	
5217	031600	044440	020123	047516	
5218	031606	020124	047503	051122	
5219	031614	041505	000124		
5220	031620	040515	047111	046440	EM45: .ASCII ?MAIN MEMORY TIMEOUT OVER THE UNIBUS DID NOT OCCUR PROPERLY?<CRLF>
5221	031626	046505	051117	020131	
5222	031634	044524	042515	052517	
5223	031642	020124	053117	051105	
5224	031650	052040	042510	052440	
5225	031656	044516	052502	020123	
5226	031664	044504	020104	047516	
5227	031672	020124	041517	052503	
5228	031700	020122	051120	050117	
5229	031706	051105	054514	200	
5230	031713	124	051505	020124	.ASCIIZ ?TEST CODE BEING RUN OVER UNIBUS?
5231	031720	047503	042504	041040	
5232	031726	044505	043516	051040	
5233	031734	047125	047440	042526	
5234	031742	020122	047125	041111	
5235	031750	051525	000		
5236	031753	122	046105	041517	EM46: .ASCII ?RELOCATION THRU THE MAP WAS NOT CORRECT, FULL ADD?<CRLF>
5237	031760	052101	047511	020116	
5238	031766	044124	052522	052040	
5239	031774	042510	046440	050101	
5240	032002	053440	051501	047040	
5241	032010	052117	041440	051117	
5242	032016	042522	052103	020054	
5243	032024	052506	046114	040440	
5244	032032	042104	200		
5245	032035	124	051505	020124	.ASCIIZ ?TEST CODE BEING RUN OVER UNIBUS?
5246	032042	047503	042504	041040	
5247	032050	044505	043516	051040	
5248	032056	047125	047440	042526	
5249	032064	020122	047125	041111	
5250	032072	051525	000		
5251	032075	122	046105	041517	EM47: .ASCII ?RELOCATION THRU THE MAP WAS NOT CORRECT, CARRY PROPAGATION?<CRLF>
5252	032102	052101	047511	020116	
5253	032110	044124	052522	052040	
5254	032116	042510	046440	050101	

5255	032124	053440	051501	047040	
5256	032132	052117	041440	051117	
5257	032140	042522	052103	020054	
5258	032146	040503	051122	020131	
5259	032154	051120	050117	043501	
5260	032162	052101	047511	100116	
5261	032170	042524	052123	041440	.ASCIIZ ?TEST CODE BEING RUN OVER UNIBUS?
5262	032176	042117	020105	042502	
5263	032204	047111	020107	052522	
5264	032212	020116	053117	051105	
5265	032220	052440	044516	052502	
5266	032226	000123			
5267	032230	044124	020105	047524	EM50: .ASCII ?THE TOP OF MEMORY IS DIFFERENT THAN THE SIZE JUMPERS?<CRLF>
5268	032236	020120	043117	046440	
5269	032244	046505	051117	020131	
5270	032252	051511	042040	043111	
5271	032260	042506	042522	052116	
5272	032266	052040	040510	020116	
5273	032274	044124	020105	044523	
5274	032302	042532	045040	046525	
5275	032310	042520	051522	200	
5276	032315	124	051505	020124	.ASCIIZ ?TEST CODE BEING RUN OVER UNIBUS?
5277	032322	047503	042504	041040	
5278	032330	044505	043516	051040	
5279	032336	047125	047440	042526	
5280	032344	020122	047125	041111	
5281	032352	051525	000		
5282	032355	120	051101	052111	EM51: .ASCII ?PARITY REPORTING THRU THE MAP IS NOT CORRECT?<CRLF>
5283	032362	020131	042522	047520	
5284	032370	052122	047111	020107	
5285	032376	044124	052522	052040	
5286	032404	042510	046440	050101	
5287	032412	044440	020123	047516	
5288	032420	020124	047503	051122	
5289	032426	041505	100124		
5290	032432	042524	052123	041440	.ASCIIZ ?TEST CODE BEING RUN OVER UNIBUS?
5291	032440	042117	020105	042502	
5292	032446	047111	020107	052522	
5293	032454	020116	053117	051105	
5294	032462	052440	044516	052502	
5295	032470	000123			
5296	032472	052523	046515	051101	EM52: .ASCIIZ ?SUMMARY OF DUAL MAPPING ERRORS?
5297	032500	020131	043117	042040	
5298	032506	040525	020114	040515	
5299	032514	050120	047111	020107	
5300	032522	051105	047522	051522	
5301	032530	000			
5302	032531	102	050131	041040	EM53: .ASCIIZ /BYP BIT IN UBMR COULD NOT BE CLEARED/
5303	032536	052111	044440	020116	
5304	032544	041125	051115	041440	
5305	032552	052517	042114	047040	
5306	032560	052117	041040	020105	
5307	032566	046103	040505	042522	
5308	032574	000104			
5309	032576	054502	020120	044502	EM54: .ASCIIZ /BYP BIT IN UBMR COULD NOT BE SET/
5310	032604	020124	047111	052440	

5311	032612	046502	020122	047503	
5312	032620	046125	020104	047516	
5313	032626	020124	042502	051440	
5314	032634	052105	000		
5315	032637	124	040522	020120	EM55: .ASCIIZ ?TRAP ON MK11 CSR ACCESS CAUSED UNIBUS ERROR?
5316	032644	047117	046440	030513	
5317	032652	020061	051503	020122	
5318	032660	041501	042503	051523	
5319	032666	041440	052501	042523	
5320	032674	020104	047125	041111	
5321	032702	051525	042440	051122	
5322	032710	051117	000		
5323	032713	124	051505	020124	EM56: .ASCIIZ ?TEST DATA REFERENCE NOT A HIT?
5324	032720	040504	040524	051040	
5325	032726	043105	051105	047105	
5326	032734	042503	047040	052117	
5327	032742	040440	044040	052111	
5328	032750	000			
5329	032751	124	051505	020124	EM57: .ASCII ?TEST DATA REFERENCE NOT A MISS?<15><12>
5330	032756	040504	040524	051040	
5331	032764	043105	051105	047105	
5332	032772	042503	047040	052117	
5333	033000	040440	046440	051511	
5334	033006	006523	012		
5335	033011	103	041501	042510	.ASCIIZ ?CACHE BYPASS ON UNIBUS MAP PAGE DID NOT INVALIDATE CACHED DATA?
5336	033016	041040	050131	051501	
5337	033024	020123	047117	052440	
5338	033032	044516	052502	020123	
5339	033040	040515	020120	040520	
5340	033046	042507	042040	042111	
5341	033054	047040	052117	044440	
5342	033062	053116	046101	042111	
5343	033070	052101	020105	040503	
5344	033076	044103	042105	042040	
5345	033104	052101	000101		
5346	033110	044124	020105	047506	EM201: .ASCIIZ ?THE FOLLOWING REGISTERS TIMED OUT WHEN READ?
5347	033116	046114	053517	047111	
5348	033124	020107	042522	044507	
5349	033132	052123	051105	020123	
5350	033140	044524	042515	020104	
5351	033146	052517	020124	044127	
5352	033154	047105	051040	040505	
5353	033162	000104			
5354	033164	044124	020105	047506	EM202: .ASCIIZ ?THE FOLLOWING MAP REGISTERS WILL NOT CLEAR?
5355	033172	046114	053517	047111	
5356	033200	020107	040515	020120	
5357	033206	042522	044507	052123	
5358	033214	051105	020123	044527	
5359	033222	046114	047040	052117	
5360	033230	041440	042514	051101	
5361	033236	000			
5362	033237	124	042510	043040	EM203: .ASCIIZ ?THE FOLLOWING ARE DUAL ADDRESSING ERRORS IN THE UNIBUS MAP?
5363	033244	046117	047514	044527	
5364	033252	043516	040440	042522	
5365	033260	042040	040525	020114	
5366	033266	042101	051104	051505	

5367	033274	044523	043516	042440
5368	033302	051122	051117	020123
5369	033310	047111	052040	042510
5370	033316	052440	044516	052502
5371	033324	020123	040515	000120
5372	033332	044124	020105	047503
5373	033340	047125	020124	040520
5374	033346	052124	051105	020116
5375	033354	044124	052522	052040
5376	033362	042510	046440	050101
5377	033370	051040	043505	051511
5378	033376	042524	051522	043040
5379	033404	044501	042514	000104
5380	033412	047125	041111	051525
5381	033420	042040	052101	020101
5382	033426	040520	044124	041440
5383	033434	052517	052116	050040
5384	033442	052101	042524	047122
5385	033450	043040	044501	052514
5386	033456	042522	000	
5387	033461	125	044516	052502
5388	033466	020123	042101	051104
5389	033474	051505	044523	043516
5390	033502	042440	051122	051117
5391	033510	026123	046440	050101
5392	033516	051040	046105	041517
5393	033524	052101	047511	020116
5394	033532	044504	040523	046102
5395	033540	042105	000	
5396	033543	103	052517	052116
5397	033550	050040	052101	042524
5398	033556	047122	043040	044501
5399	033564	052514	042522	020123
5400	033572	047111	041440	041501
5401	033600	042510	051040	043505
5402	033606	051511	042524	051522
5403	033614	000		
5404				
5405	033615	122	041505	044505
5406	033622	042126	042440	050130
5407	033630	041505	042124	052040
5408	033636	051505	047124	020117
5409	033644	050040	020103	052101
5410	033652	040440	047502	052122
5411	033660	000		
5412	033661	122	041505	044505
5413	033666	042126	052040	051505
5414	033674	047124	020117	050040
5415	033702	020103	052101	040440
5416	033710	047502	052122	000
5417	033715	103	047117	044504
5418	033722	047124	040440	042104
5419	033730	042522	051523	020040
5420	033736	046440	044501	052116
5421	033744	047105	041440	047117
5422	033752	051124	046117	200

EM204: .ASCIZ ?THE COUNT PATTERN THRU THE MAP REGISTERS FAILED?

EM205: .ASCIZ ?UNIBUS DATA PATH COUNT PATTERN FAILURE?

EM206: .ASCIZ ?UNIBUS ADDRESSING ERRORS, MAP RELOCATION DISABLED?

EM207: .ASCIZ ?COUNT PATTERN FAILURES IN CACHE REGISTERS?

DH1: .ASCIZ ?RECEIVD EXPECTD TESTNO PC AT ABORT?

DH2: .ASCIZ ?RECEIVD TESTNO PC AT ABORT?

DH3: .ASCII ?CONDITN ADDRESS MAINTEN CONTROL?<CRLF>

5423	033757	122	041505	044505
5424	033764	042126	051040	043105
5425	033772	051105	047105	042103
5426	034000	051040	043505	051511
5427	034006	051124	051040	043505
5428	034014	051511	051124	052040
5429	034022	051505	047124	020117
5430	034030	050040	020103	052101
5431	034036	040440	047502	052122
5432	034044	000		
5433	034045	123	040524	052524
5434	034052	020123	040440	052125
5435	034060	044517	042057	053040
5436	034066	051111	052524	046101
5437	034074	200		
5438	034075	122	043505	051511
5439	034102	051124	051040	043505
5440	034110	051511	051124	040440
5441	034116	042104	042522	051523
5442	034124	052040	051505	047124
5443	034132	020117	050040	020103
5444	034140	052101	040440	047502
5445	034146	052122	000	
5446	034151	122	043505	042101
5447	034156	051522	051040	043505
5448	034164	042101	051522	200
5449	034171	040	047442	021122
5450	034176	020040	020040	040442
5451	034204	042116	020042	021440
5452	034212	051105	047522	051522
5453	034220	052040	051505	047124
5454	034226	020117	042440	051122
5455	034234	051117	041520	000
5456	034241	122	043505	042101
5457	034246	051522	051040	043505
5458	034254	042101	051522	051040
5459	034262	041505	044505	042126
5460	034270	051040	041505	044505
5461	034276	042126	200	
5462	034301	040	047442	021122
5463	034306	020040	020040	040442
5464	034314	042116	020042	020040
5465	034322	047442	021122	020040
5466	034330	020040	040442	042116
5467	034336	020042	021440	051105
5468	034344	047522	051522	052040
5469	034352	051505	047124	020117
5470	034360	042440	051122	051117
5471	034366	041520	000	
5472	034371	103	052517	052116
5473	034376	020040	041440	052517
5474	034404	052116	200	
5475	034407	105	050130	041505
5476	034414	042124	051040	041505
5477	034422	044505	042126	052040
5478	034430	051505	047124	020117

.DSCIZ ?RECEIVD REFERENC'D REGISTR REGISTR TESTNO PC AT ABORT?

DH5: .ASCII ?STATUS AUTOI/D VIRTUAL?<CRLF>

.DSCIZ ?REGISTR REGISTR ADDRESS TESTNO PC AT ABORT?

DH6: .ASCII ?REGADRS REGADRS?<CRLF>

.DSCIZ ? 'OR' 'AND' #ERRORS TESTNO ERRORPC?

DH10: .ASCII ?REGADRS REGADRS RECEIVD RECEIVD?<CRLF>

.DSCIZ ? 'OR' 'AND' 'OR' 'AND' #ERRORS TESTNO ERRORPC?

DH12: .ASCII ?COUNT COUNT?<CRLF>

.DSCIZ ?EXPECTD RECEIVD TESTNO ERRORPC?

5479	034436	042440	051122	051117	
5480	034444	041520	000		
5481	034447	122	043505	047514	DH14: .ASCII ?REGLOAD REGLOAD REGDUAL REGDUAL?<CRLF>
5482	034454	042101	051040	043505	
5483	034462	047514	042101	051040	
5484	034470	043505	052504	046101	
5485	034476	051040	043505	052504	
5486	034504	046101	200		
5487	034507	040	047442	021122	.ASCIIZ ? 'OR' 'AND' 'OR' 'AND' #ERRORS TESTNO?
5488	034514	020040	020040	040442	
5489	034522	042116	020042	020040	
5490	034530	047442	021122	020040	
5491	034536	020040	040442	042116	
5492	034544	020042	021440	051105	
5493	034552	047522	051522	052040	
5494	034560	051505	047124	000117	
5495	034566	040515	051120	043505	DH15: .ASCII ?MAPREG MAPREG EXPECTD EXPECTD RECEIVD RECEIVD?<CRLF>
5496	034574	020040	040515	051120	
5497	034602	043505	020040	054105	
5498	034610	042520	052103	020104	
5499	034616	054105	042520	052103	
5500	034624	020104	042522	042503	
5501	034632	053111	020104	042522	
5502	034640	042503	053111	100104	
5503	034646	021040	051117	020042	.ASCIIZ ? 'OR' 'AND' 'OR' 'AND' 'OR' 'AND' #ERRORS TESTNO?
5504	034654	020040	021040	047101	
5505	034662	021104	020040	021040	
5506	034670	051117	020042	020040	
5507	034676	021040	047101	021104	
5508	034704	020040	021040	051117	
5509	034712	020042	020040	021040	
5510	034720	047101	021104	020040	
5511	034726	042443	051122	051117	
5512	034734	020123	042524	052123	
5513	034742	047516	000		
5514	034745	122	041505	044505	DH17: .ASCIIZ ?RECEIVD TESTNO ERRORPC?
5515	034752	042126	052040	051505	
5516	034760	047124	020117	042440	
5517	034766	051122	051117	041520	
5518	034774	000			
5519	034775	105	050130	041505	DH23: .ASCII ?EXPECTD EXPECTD RECEIVD RECEIVD?<CRLF>
5520	035002	042124	042440	050130	
5521	035010	041505	042124	051040	
5522	035016	041505	044505	042126	
5523	035024	051040	041505	044505	
5524	035032	042126	200		
5525	035035	040	047442	021122	.ASCIIZ ? 'OR' 'AND' 'OR' 'AND' #ERRORS TESTNO?
5526	035042	020040	020040	040442	
5527	035050	042116	020042	020040	
5528	035056	047442	021122	020040	
5529	035064	020040	040442	042116	
5530	035072	020042	021440	051105	
5531	035100	047522	051522	052040	
5532	035106	051505	047124	000117	
5533	035114	042101	051104	051525	DH25: .ASCIIZ ?ADDRUSED BITDIFF TESTNO ERRORPC?
5534	035122	042105	020040	044502	

5535	035130	042124	043111	020106	
5536	035136	042524	052123	047516	
5537	035144	020040	051105	047522	
5538	035152	050122	000103		
5539	035156	042101	051104	051525	DH27: .ASCIZ ?ADDRUSED TESTNO ERRORPC?
5540	035164	042105	020040	042524	
5541	035172	052123	047516	020040	
5542	035200	051105	047522	050122	
5543	035206	000103			
5544	035210	042524	052123	047516	DH30: .ASCIZ ?TESTNO ERRORPC?
5545	035216	020040	051105	047522	
5546	035224	050122	000103		
5547	035230	042522	047515	042526	DH34: .ASCIZ ?REMOVED MISSING TESTNO ERRORPC?
5548	035236	020104	044515	051523	
5549	035244	047111	020107	042524	
5550	035252	052123	047516	020040	
5551	035260	051105	047522	050122	
5552	035266	000103			
5553	035270	047514	042527	052123	DH35: .ASCIZ ?LOWEST HIGHEST TESTNO ERRORPC?
5554	035276	020040	044510	044107	
5555	035304	051505	020124	042524	
5556	035312	052123	047516	020040	
5557	035320	051105	047522	050122	
5558	035326	000103			
5559	035330	042524	052123	047516	DH36: .ASCIZ ?TESTNO ERRORPC UNIBUS ADDRESS OF MAP REGISTER UNDER TEST?
5560	035336	020040	051105	047522	
5561	035344	050122	020103	047125	
5562	035352	041111	051525	040440	
5563	035360	042104	042522	051523	
5564	035366	047440	020106	040515	
5565	035374	020120	042522	044507	
5566	035402	052123	051105	052440	
5567	035410	042116	051105	052040	
5568	035416	051505	000124		
5569	035422	047503	042116	052111	DH40: .ASCII ?CONDITN CONDITN?<CRLF>
5570	035430	020116	047503	042116	
5571	035436	052111	100116		
5572	035442	054105	042520	052103	.ASCIZ ?EXPECTED RECEIVD TESTNO ERRORPC?
5573	035450	020104	042522	042503	
5574	035456	053111	020104	042524	
5575	035464	052123	047516	020040	
5576	035472	051105	047522	050122	
5577	035500	000103			
5578	035502	047503	051122	041505	DH41: .ASCII ?CORRECT ADDRESS?<CRLF>
5579	035510	020124	042101	051104	
5580	035516	051505	100123		
5581	035522	042101	051104	051505	.ASCIZ ?ADDRESS FETCHED TESTNO ERRORPC?
5582	035530	020123	042506	041524	
5583	035536	042510	020104	042524	
5584	035544	052123	047516	020040	
5585	035552	051105	047522	050122	
5586	035560	000103			
5587	035562	047503	051122	041505	DH42: .ASCII ?CORRECT EXPECTD RECEIVD?<CRLF>
5588	035570	020124	054105	042520	
5589	035576	052103	020104	042522	
5590	035604	042503	053111	100104	

5591	035612	042101	051104	051505		.ASCIZ	?ADDRESS DATA	FROM UB TESTNO	ERRORPC?
5592	035620	020123	040504	040524					
5593	035626	020040	020040	051106					
5594	035634	046517	052440	020102					
5595	035642	042524	052123	047516					
5596	035650	020040	051105	047522					
5597	035656	050122	000103						
5598	035662	044523	045132	046525	DH43:	.ASCIZ	?SIZJUMP TOPFOUND	TESTNO	ERRORPC?
5599	035670	020120	047524	043120					
5600	035676	052517	042116	052040					
5601	035704	051505	047124	020117					
5602	035712	051105	047522	050122					
5603	035720	000103							
5604	035722	047503	042116	052111	DH44:	.ASCII	?CONDITN CONDITN ADDRESS	MAINTEN CONTROL?	<CRLF>
5605	035730	020116	047503	042116					
5606	035736	052111	020116	042101					
5607	035744	051104	051505	020123					
5608	035752	020040	040515	047111					
5609	035760	042524	020116	047503					
5610	035766	052116	047522	100114					
5611	035774	054105	042520	052103		.ASCIZ	?EXPECTD RECEIVD	REFERENC D REGISTR	REGISTR TESTNO ERRORPC?
5612	036002	020104	042522	042503					
5613	036010	053111	020104	042522					
5614	036016	042506	042522	041516					
5615	036024	020104	042522	044507					
5616	036032	052123	020122	042522					
5617	036040	044507	052123	020122					
5618	036046	042524	052123	047516					
5619	036054	020040	051105	047522					
5620	036062	050122	000103						
5621	036066	020040	020040	041520	DH53:	.ASCIZ	/ PC	UBMR (UBMR)/	
5622	036074	020040	020040	041125					
5623	036102	051115	020040	052450					
5624	036110	046502	024522	000					
5625	036115	105	051122	051117	DH55:	.ASCIZ	/ERRORPC	MKCSR	MSER CPUERR/
5626	036122	041520	020040	045515					
5627	036130	051503	020122	020040					
5628	036136	020040	051515	051105					
5629	036144	020040	020040	041440					
5630	036152	052520	051105	000122					
5631	036160	020040	041520	020040	DH56:	.ASCIZ	/ PC	CCR GROUP	TST-DATA-ADRS/
5632	036166	020040	041440	051103					
5633	036174	020040	043440	047522					
5634	036202	050125	020040	052040					
5635	036210	052123	042055	052101					
5636	036216	026501	042101	051522					
5637	036224	000							
5638	036225	040	050040	020103	DH57:	.ASCIZ	/ PC	CCR	UBMR TST-DATA-ADRS/
5639	036232	020040	041440	051103					
5640	036240	020040	020040	041125					
5641	036246	051115	020040	052040					
5642	036254	052123	042055	052101					
5643	036262	026501	042101	051522					
5644	036270	000							
5645	036271	122	043505	042101	DH201:	.ASCIZ	?REGADRS	TESTNO	ERRORPC?
5646	036276	051522	052040	051505					

5647	036304	047124	020117	042440	
5648	036312	051122	051117	041520	
5649	036320	000			
5650	036321	122	043505	042101	DH202: .ASCIZ ?REGADRS DATAREC TESTNO ERRORPC?
5651	036326	051522	042040	052101	
5652	036334	051101	041505	052040	
5653	036342	051505	047124	020117	
5654	036350	042440	051122	051117	
5655	036356	041520	000		
5656	036361	115	050101	042522	DH203: .ASCII ?MAPREG MAPREG?<CRLF>
5657	036366	020107	046440	050101	
5658	036374	042522	100107		
5659	036400	042524	052123	047111	.ASCIZ ?TESTING DUALED TESTNO ERRORPC?
5660	036406	020107	052504	046101	
5661	036414	042105	020040	042524	
5662	036422	052123	047516	020040	
5663	036430	051105	047522	050122	
5664	036436	000103			
5665	036440	042522	040507	051104	DH204: .ASCIZ ?REGADRS PATTERN EXPECTD RECEIVD TESTNO ERRORPC?
5666	036446	020123	040520	052124	
5667	036454	051105	020116	054105	
5668	036462	042520	052103	020104	
5669	036470	042522	042503	053111	
5670	036476	020104	042524	052123	
5671	036504	047516	020040	051105	
5672	036512	047522	050122	000103	
5673	036520	054105	042520	052103	DH205: .ASCIZ ?EXPECTD RECEIVD ADDRSLOAD TESTNO ERRORPC?
5674	036526	020104	042522	042503	
5675	036534	053111	020104	042101	
5676	036542	051104	046123	040517	
5677	036550	020104	042524	052123	
5678	036556	047516	020040	051105	
5679	036564	047522	050122	000103	
5680	036572	042101	051104	051505	DH206: .ASCII ?ADDRESS ADDRESS?<CRLF>
5681	036600	020123	042101	051104	
5682	036606	051505	100123		
5683	036612	054105	042520	052103	.ASCIZ ?EXPECTD RECEIVD TESTNO ERRORPC?
5684	036620	020104	042522	042503	
5685	036626	053111	020104	042524	
5686	036634	052123	047516	020040	
5687	036642	051105	047522	050122	
5688	036650	000103			
5689	036652	042522	044507	052123	DH207: .ASCII ?REGISTR EXPECTD RECEIVD?<CRLF>
5690	036660	020122	054105	042520	
5691	036666	052103	020104	042522	
5692	036674	042503	053111	100104	
5693	036702	042101	051104	051505	.ASCIZ ?ADDRESS DATA DATA TESTNO ERRORPC?
5694	036710	020123	042040	052101	
5695	036716	020101	020040	042040	
5696	036724	052101	020101	020040	
5697	036732	042524	052123	047516	
5698	036740	020040	051105	047522	
5699	036746	050122	000103		
5700					
5701					
5702					.EVEN

5703	036752	001266	001264	001262	DT1:	.WORD	PCPUER,CPUEXP,TESTNO,BADPC,0
5704	036760	001302	000000				
5705	036764	001266	001262	001302	DT2:	.WORD	PCPUER,TESTNO,BADPC,0
5706	036772	000000					
5707	036774	001274	001270	001300	DT3:	.WORD	PPARER,PLOADR,PMaint,PCONTR,TESTNO,BADPC,0
5708	037002	001276	001262	001302			
5709	037010	000000					
5710	037012	001154	001170	001262	DT4:	.WORD	\$REG0,\$TMP0,TESTNO,\$ERRPC,0
5711	037020	001116	000000				
5712	037024	001312	001314	001316	DT5:	.WORD	PMMR0,PMMR1,PMMR2,TESTNO,BADPC,0
5713	037032	001262	001302	000000			
5714	037040	001226	001224	001254	DT6:	.WORD	ADDROR,ADRAND,ERRCNT,TESTNO,\$ERRPC,0
5715	037046	001262	001116	000000			
5716	037054	001226	001224	001232	DT10:	.WORD	ADDROR,ADRAND,DATAOR,DATAND,ERRCNT,TESTNO,\$ERRPC,0
5717	037062	001230	001254	001262			
5718	037070	001116	000000				
5719	037074	001160	001154	001262	DT12:	.WORD	\$REG2,\$REG0,TESTNO,\$ERRPC,0
5720	037102	001116	000000				
5721	037106	001162	001156	001262	DT13:	.WORD	\$REG3,\$REG1,TESTNO,\$ERRPC,0
5722	037114	001116	000000				
5723	037120	001226	001224	001232	DT14:	.WORD	ADDROR,ADRAND,DATAOR,DATAND,ERRCNT,TESTNO,0
5724	037126	001230	001254	001262			
5725	037134	000000					
5726	037136	001226	001224	001236	DT15:	.WORD	ADDROR,ADRAND,PATTOR,PATAND,DATAOR,DATAND,ERRCNT,TESTNO,0
5727	037144	001234	001232	001230			
5728	037152	001254	001262	000000			
5729	037160	001154	001262	001116	DT17:	.WORD	\$REG0,TESTNO,\$ERRPC,0
5730	037166	000000					
5731	037170	001156	001262	001116	DT20:	.WORD	\$REG1,TESTNO,\$ERRPC,0
5732	037176	000000					
5733	037200	001236	001234	001232	DT23:	.WORD	PATTOR,PATAND,DATAOR,DATAND,ERRCNT,TESTNO,0
5734	037206	001230	001254	001262			
5735	037214	000000					
5736	037216	001164	001154	001262	DT25:	.WORD	\$REG4,\$REG0,TESTNO,\$ERRPC,0
5737	037224	001116	000000				
5738	037230	001156	001262	001116	DT27:	.WORD	\$REG1,TESTNO,\$ERRPC,0
5739	037236	000000					
5740	037240	001262	001116	000000	DT30:	.WORD	TESTNO,\$ERRPC,0
5741	037246	001254	001256	001262	DT34:	.WORD	ERRCNT,CNTR,TESTNO,\$ERRPC,0
5742	037254	001116	000000				
5743	037260	001240	001242	001262	DT35:	.WORD	LOWEST,HIGEST,TESTNO,\$ERRPC,0
5744	037266	001116	000000				
5745	037272	001262	001116	001154	DT36:	.WORD	TESTNO,\$ERRPC,\$REG0,0
5746	037300	000000					
5747	037302	001226	001224	001232	DT37:	.WORD	ADDROR,ADRAND,DATAOR,DATAND,ERRCNT,TESTNO,0
5748	037310	001230	001254	001262			
5749	037316	000000					
5750	037320	001264	001266	001262	DT40:	.WORD	CPUEXP,PCPUER,TESTNO,\$ERRPC,0
5751	037326	001116	000000				
5752	037332	001160	001156	001262	DT41:	.WORD	\$REG2,\$REG1,TESTNO,\$ERRPC,0
5753	037340	001116	000000				
5754	037344	001156	001162	001160	DT42:	.WORD	\$REG1,\$REG3,\$REG2,TESTNO,\$ERRPC,0
5755	037352	001262	001116	000000			
5756	037360	177760	001320	001262	DT43:	.WORD	SIZELO,RSIZE,TESTNO,\$ERRPC,0
5757	037366	001116	000000				
5758	037372	001200	001274	001270	DT44:	.WORD	\$TMP4,PPARER,PLOADR,PMaint,PCONTR,TESTNO,\$ERRPC,0

5759	037400	001300	001276	001262			
5760	037406	001116	000000				
5761	037412	001116	001154	001156	DT53:	.WORD	\$ERRPC,\$REG0,\$REG1,0
5762	037420	000000					
5763	037422	001262	001160	001172	DT55:	.WORD	TESTNO,\$REG2,\$TMP1,\$TMP2,0
5764	037430	001174	000000				
5765	037434	001116	001154	001156	DT56:	.WORD	\$ERRPC,\$REG0,\$REG1,\$REG2,0
5766	037442	001160	000000				
5767	037446	001116	001154	001156	DT57:	.WORD	\$ERRPC,\$REG0,\$REG1,\$REG2,0
5768	037454	001160	000000				
5769	037460	001154	001262	001116	DT201:	.WORD	\$REG0,TESTNO,\$ERRPC,0
5770	037466	000000					
5771	037470	001154	001170	001262	DT202:	.WORD	\$REG0,\$TMP0,TESTNO,\$ERRPC,0
5772	037476	001116	000000				
5773	037502	001154	001156	001262	DT203:	.WORD	\$REG0,\$REG1,TESTNO,\$ERRPC,0
5774	037510	001116	000000				
5775	037514	001154	001160	001164	DT204:	.WORD	\$REG0,\$REG2,\$REG4,\$REG3,TESTNO,\$ERRPC,0
5776	037522	001162	001262	001116			
5777	037530	000000					
5778	037532	001156	001154	001160	DT205:	.WORD	\$REG1,\$REG0,\$REG2,TESTNO,\$ERRPC,0
5779	037540	001262	001116	000000			
5780	037546	001154	001162	001262	DT206:	.WORD	\$REG0,\$REG3,TESTNO,\$ERRPC,0
5781	037554	001116	000000				
5782	037560	001154	001160	001162	DT207:	.WORD	\$REG0,\$REG2,\$REG3,TESTNO,\$ERRPC,0
5783	037566	001262	001116	000000			
5784							
5785							
5786	037574	000	000	000	DF1:	.BYTE	0,0,0,0
5787	037577	000					
5788	037600	000	000	000	DF2:	.BYTE	0,0,0
5789	037603	000	002	000	DF3:	.BYTE	0,2,0,0,0,0
5790	037606	000	000	000			
5791	037611	000	000	000	DF5:	.BYTE	0,0,0,0,0
5792	037614	000	000				
5793	037616	000	000	001	DF6:	.BYTE	0,0,1,0,0
5794	037621	000	000				
5795	037623	000	000	000	DF10:	.BYTE	0,0,0,0,1,0,0
5796	037626	000	001	000			
5797	037631	000					
5798	037632	000	000	000	DF12:	.BYTE	0,0,0,0
5799	037635	000					
5800	037636	000	000	000	DF14:	.BYTE	0,0,0,0,1,0
5801	037641	000	001	000			
5802	037644	000	000	000	DF15:	.BYTE	0,0,0,0,0,0,1,0
5803	037647	000	000	000			
5804	037652	001	000				
5805	037654	000	000	000	DF17:	.BYTE	0,0,0
5806	037657	000	000	000	DF23:	.BYTE	0,0,0,0,1,0
5807	037662	000	001	000			
5808	037665	003	004	000	DF25:	.BYTE	3,4,0,0
5809	037670	000					
5810	037671	003	000	000	DF27:	.BYTE	3,0,0
5811	037674	000	000		DF30:	.BYTE	0,0
5812	037676	000	000	000	DF34:	.BYTE	0,0,0,0
5813	037701	000					
5814	037702	004	004	000	DF35:	.BYTE	4,4,0,0

5815	037705	000				
5816	037706	000	000	003	DF36: .BYTE	0,0,3
5817	037711	000	000	000	DF37: .BYTE	0,0,0,0,1,0
5818	037714	000	001	000		
5819	037717	000	000	000	DF40: .BYTE	0,0,0,0
5820	037722	000				
5821	037723	000	000	000	DF41: .BYTE	0,0,0,0
5822	037726	000				
5823	037727	003	000	000	DF42: .BYTE	3,0,0,0,0
5824	037732	000	000			
5825	037734	000	000	000	DF43: .BYTE	0,0,0,0
5826	037737	000				
5827	037740	000	000	002	DF44: .BYTE	0,0,2,0,0,0,0
5828	037743	000	000	000		
5829	037746	000				
5830	037747	000	000	000	DF201: .BYTE	0,0,0
5831	037752	000	000	000	DF202: .BYTE	0,0,0,0
5832	037755	000				
5833	037756	000	000	000	DF203: .BYTE	0,0,0,0
5834	037761	000				
5835	037762	000	000	000	DF204: .BYTE	0,0,0,0,0,0
5836	037765	000	000	000		
5837	037770	000	000	003	DF205: .BYTE	0,0,3,0,0
5838	037773	000	000			
5839	037775	000	000	000	DF206: .BYTE	0,0,0,0
5840	040000	000				
5841	040001	000	000	000	DF207: .BYTE	0,0,0,0,0
5842	040004	000	000			
5843						
5844						
5845	040006	000017			SAVEA: .EVEN	
5846					.BLKW	17
5847		040044			TSLOC=.	
5848		040044			TSLOC=-4&TSLOC	
5849		040050			TSLOC=TSLOC+4	
5850		040050			.=TSLOC	
5851						
5852	040050	001000			TSTDAT: .BLKW	512.
5853		000001			.END	

;GET PC TO AN EVEN WORD BOUNDARY

ADDROR	001226	519#	987*	1755*	1781*	1808*	1834*	2855*	3324*	5714	5716	5723	5726	5747
ADRAND	001224	518#	991*	1757*	1783*	1810*	1836*	2859*	3319*	5714	5716	5723	5726	5747
BADPC	001302	547#	1939*	1985*	2031*	5703	5705	5707	5712					
BIT0 =	000001	133#	2121	2122	2129	2909								
BIT00 =	000001	123#	133											
BIT01 =	000002	122#	132											
BIT02 =	000004	121#	131											
BIT03 =	000010	120#	130											
BIT04 =	000020	119#	129											
BIT05 =	000040	118#	128											
BIT06 =	000100	117#	127											
BIT07 =	000200	116#	126											
BIT08 =	000400	115#	125	1007										
BIT09 =	001000	114#	124	1017	1090									
BIT1 =	000002	132#	1994*											
BIT10 =	002000	113#	1073											
BIT11 =	004000	112#	1024	2124	2125	2127	2935							
BIT12 =	010000	111#	417	4750										
BIT13 =	020000	110#	1080											
BIT14 =	040000	109#	2116	2117	2119									
BIT15 =	100000	108#	416	2130	2131	2133	2135	2136	2138	3774	3845	4369	4442	
BIT2 =	000004	131#	420	421	3763	4358								
BIT3 =	000010	130#	420	421	3834	4431								
BIT4 =	000020	129#	419	421	1686	2910	2929	4549						
BIT5 =	000040	128#	418	420	2974	3144	3281	3351	3571					
BIT6 =	000100	127#												
BIT7 =	000200	126#												
BIT8 =	000400	125#	2151											
BIT9 =	001000	124#	1143											
BPTVEC=	000014	140#												
BYP =	100000	416#	4495	4501	4502	4508	4509	4645	4659					
CACHE=	000114	147#	1976*	1988*	1997*	2177*	2179*	3661*	3672*	3718*	3729*	3781*	3800*	3852*
		3871*	4250*	4261*	4310*	4321*	4376*	4395*	4449*	4468*				
CACOUN	005576	1892#	2844	2874										
CBMP	023702	4579	4581	4583#	4592									
CBMPA	023722	4588#	4687											
CISP	001333	562#												
CLRMAP	005134	1723#	2547	3033	3230									
CNTR	001256	537#	3172*	3176	3180*	5741								
CONTRL=	177746	157#	1983	2113	2143*	2144	2774	2778*	2779	2833	2834	2836	2861*	2881*
		3652	3653*	3673*	3681	3707	3708*	3730*	3738	3763	3765	3778*	3801*	3809
		3834	3836	3849*	3872*	3880	4241	4242*	4262*	4270	4299	4300*	4322*	4330
		4358	4360	4373*	4396*	4404	4431	4433	4446*	4469*	4477	4588*	4589	4594*
		4596*	4602*	4632	4667									
COUNT	005432	1833#	2600	2644	2689	2733								
CPFLAG	005654	1103*	1927#	1948*	2193*									
CPUER	005652	1926#	2183	2291										
CPUErr=	177766	170#	1099*	1749	1750*	1938	1947*	2198*	4549	4553	4555*			
CPUEXP	001264	540#	1751	1940	1942	2189*	2930*	2953*	2970*	3008*	3034*	3072*	3093*	3111*
		3138*	3181*	3352*	3370*	3581*	3592*	3626*	3936*	3954*	4170*	4181*	4215*	5703
		5750												
CR =	000015	45#	1393	1403										
CRLF =	000200	46#	1366	1403	2091	4766	4772	4774	4776	4781	4928	4943	4959	4977
		5029	5043	5081	5100	5118	5127	5137	5220	5236	5251	5267	5282	5417
		5433	5446	5456	5472	5481	5495	5519	5569	5578	5587	5604	5656	5680
		5689												

[illegible]

DH36	035330	790	5559#			
DH40	035422	803	838	5569#		
DH41	035502	810	846	5578#		
DH42	035562	817	854	5587#		
DH43	035662	824	862	5598#		
DH44	035722	830	870	5604#		
DH5	034045	613	5433#			
DH53	036066	885	892	5621#		
DH55	036115	898	5625#			
DH56	036160	905	5631#			
DH57	036225	912	5638#			
DH6	034151	620	627	5446#		
DISPLA=	177570	40#	1038*	1069*	2928*	3137* 3934*
DT1	036752	586	5703#			
DT10	037054	636	643	5716#		
DT12	037074	650	5719#			
DT13	037106	657	5721#			
DT14	037120	664	5723#			
DT15	037136	671	678	5726#		
DT17	037160	685	699	706	5729#	
DT2	036764	592	5705#			
DT20	037170	692	5731#			
DT201	037460	923	5769#			
DT202	037470	929	5771#			
DT203	037502	936	5773#			
DT204	037514	942	5775#			
DT205	037532	948	5778#			
DT206	037546	955	5780#			
DT207	037560	962	5782#			
DT23	037200	713	720	755	5733#	
DT25	037216	727	734	5736#		
DT27	037230	741	5738#			
DT3	036774	600	608	5707#		
DT30	037240	748	761	769	5740#	
DT34	037246	776	5741#			
DT35	037260	785	5743#			
DT36	037272	791	5745#			
DT37	037302	798	879	5747#		
DT4	037012	5710#				
DT40	037320	805	840	5750#		
DT41	037332	812	848	5752#		
DT42	037344	819	856	5754#		
DT43	037360	825	863	5756#		
DT44	037372	832	872	5758#		
DT5	037024	615	5712#			
DT53	037412	886	893	5761#		
DT55	037422	899	5763#			
DT56	037434	906	5765#			
DT57	037446	913	5767#			
DT6	037040	622	629	5714#		
DUALAD	005356	1806#	2555	3246		
EMTVEC=	000030	143#	2060*	2061*		
EM1	024773	584	4786#			
EM10	025604	633	4854#			
EM11	025675	640	4864#			
EM12	025767	647	4874#			

EM13	026055	654	4884#
EM14	026147	661	4894#
EM15	026242	668	4904#
EM16	026346	675	4916#
EM17	026451	682	4928#
EM2	025055	590	4795#
EM20	026573	689	4943#
EM201	033110	921	5346#
EM202	033164	927	5354#
EM203	033237	933	5362#
EM204	033332	940	5372#
EM205	033412	946	5380#
EM206	033461	952	5387#
EM207	033543	959	5396#
EM21	026721	696	4959#
EM22	027065	703	4977#
EM23	027231	710	4995#
EM24	027325	717	5006#
EM25	027425	724	5017#
EM26	027532	731	738
EM3	025124	596	4802#
EM30	027653	745	5043#
EM31	030027	752	5063#
EM32	030121	759	5073#
EM33	030173	765	5081#
EM34	030350	773	5100#
EM35	030521	780	5118#
EM36	031006	789	5151#
EM37	031103	795	5162#
EM4	025225	604	4813#
EM40	031206	802	5174#
EM41	031301	809	5184#
EM42	031363	816	5193#
EM43	031456	823	5203#
EM44	031543	829	5212#
EM45	031620	836	5220#
EM46	031753	844	5236#
EM47	032075	852	5251#
EM5	025334	612	4825#
EM50	032230	860	5267#
EM51	032355	868	5282#
EM52	032472	876	5296#
EM53	032531	884	5302#
EM54	032576	891	5309#
EM55	032637	897	5315#
EM56	032713	904	5323#
EM57	032751	911	5329#
EM6	025442	619	4837#
EM7	025522	626	4845#
ENDKB	010614	2154	2159#
EREXIT	003034	1098	1106#
ERRCNT	001254	536#	985*
		2406	2564
		3261	3326
		136#	998
ERRVEC=	000004	1082	1133#
ERTYPE	003036		

5029#

1060*	1759	1789	1816	1846	1873	2231	2262	2304	2338	2372
2611	2655	2700	2744	2848	2853*	2877	3067	3168*	3176	3179*
3332	5714	5716	5723	5726	5733	5741	5747			
999*	1001*	1004*	2183*	2184*	2221*	2291*				

PDP-11/70-74MP UNIBUS MAP DIAGNOSTIC CEKBFC.P11 03-MAY-79 13:19			MACY11 30A(1052) 03-MAY-79 13:21 PAGE 119 CROSS REFERENCE TABLE -- USER SYMBOLS												SEQ 0129	
ER200 002124			916#	1153	1156											
FLAG 001260			538#													
GNS = ***** U			430	1574	1575	1576	1577	1578	1579	1580	1581	1582	2090	4719	4726	
HIADRS= 177742			155#	1981	2800	3679	3736	3807	3878	4268	4328	4402	4475			
HIGEST 001242			526#	3159*	3186	3193	3250	3259	5743							
HITMIS= 177752			159#	2258	4630	4665										
HREGL 001250			532#	3204*												
HREGU 001252			534#	3206*												
HT = 000011			43#	1364	1403											
IOTVEC= 000020			141#	2058*	2059*											
KBTST 010340			2105#													
KB11CM 001332			561#	2105*	2153*	2165	2487	3623	4212	4491	4533	4580				
KB11E 001330			559#	2106*	2110*	2149	2151*	2163	2171							
KB11EM 001331			560#	2485	3621	4210	4489	4531	4578							
KDPAR0= 172360			313#													
KDPAR1= 172362			314#													
KDPAR2= 172364			315#													
KDPAR3= 172366			316#													
KDPAR4= 172370			317#													
KDPAR5= 172372			318#													
KDPAR6= 172374			319#													
KDPAR7= 172376			320#													
KDPDR0= 172320			291#													
KDPDR1= 172322			292#													
KDPDR2= 172324			293#													
KDPDR3= 172326			294#													
KDPDR4= 172330			295#													
KDPDR5= 172332			296#													
KDPDR6= 172334			297#													
KDPDR7= 172336			298#													
KERSTK= 001100			30#	1936												
KIPAR0= 172340			302#	1245	2901*	3914*	4608*									
KIPAR1= 172342			303#	2902*	3912*	4610*										
KIPAR2= 172344			304#	2903*	4612*											
KIPAR3= 172346			305#	2904*												
KIPAR4= 172350			306#	2905*	3146*	3153*	3154	3158	3159	3160*	3161	3173*	3174	3240*	3282*	
			3289*	3290	3353*	3578*	3606	3608*	3937*	4167*	4195	4197*				
KIPAR5= 172352			307#	2906*	3047*	3048*										
KIPAR6= 172354			308#	2907*	2932*	2937*	2948*	2972*	2976*	2983*	3036*	3042*	3043	3047	3095	
			3297	3300	3312*	3313	3395*	3410*	3425*	3440*	3455*	3470*	3485*	3500*	3515*	
			3530*	3545*	3579*	3654*	3709*	3769*	3840*	3984*	3999*	4014*	4029*	4044*	4059*	
			4074*	4089*	4104*	4119*	4134*	4168*	4243*	4301*	4364*	4437*	4583*	4686*		
KIPAR7= 172356			309#	2908*	4606*											
KIPDR0= 172300			280#	2130*	2131	2133*	2893*	4609*								
KIPDR1= 172302			281#	2894*	4611*											
KIPDR2= 172304			282#	2895*	4613*											
KIPDR3= 172306			283#	2896*												
KIPDR4= 172310			284#	2897*												
KIPDR5= 172312			285#	2898*												
KIPDR6= 172314			286#	2899*	4584*											
KIPDR7= 172316			287#	2900*	4607*											
LF = 000012			44#	1397	1403											
LOADRS= 177740			154#	1980	2253	2793	3678	3735	3806	3877	4267	4327	4401	4474		
LOOP 010676			2177#	4760												
LOWEST 001240			524#	3158*	3184	3194	3235	3236	3353	3395	3410	3425	3440	3455	3470	
			3485	3500	3515	3530	3545	3579	3654	3709	3769	3840	3910	3937	3984	

	3999	4014	4029	4044	4059	4074	4089	4104	4119	4134	4168	4243	4301
LREGL 001244	4364	4437	5743										
	528#	3201*	3234	3355*	3394*	3409*	3424*	3439*	3454*	3469*	3484*	3499*	3514*
	3529*	3544*	3575*	3609*	3655*	3710*	3771*	3842*	3939*	3983*	3998*	4013*	4028*
LREGU 001246	4043*	4058*	4073*	4088*	4103*	4118*	4133*	4164*	4198*	4244*	4302*	4366*	4439*
	530#	3203*	3354*	3393*	3408*	3423*	3438*	3453*	3468*	3483*	3498*	3513*	3528*
	3543*	3574*	3611*	3612	3656*	3711*	3770*	3841*	3904	3938*	3982*	3997*	4012*
	4027*	4042*	4057*	4072*	4087*	4102*	4117*	4132*	4163*	4200*	4201	4245*	4303*
	4365*	4438*											
MAINT = 177750	158#	1984	2111*	2114	2786*	2787	2862	3657	3662*	3712	3719*	3772	3782*
	3843	3853*	4246	4251*	4304	4311*	4367	4377*	4440	4450*			
MAPHO = 170202	397#	2326	2493*										
MAPH00= 170202	333#	397	2115	2494	2638								
MAPH01= 170206	335#	399											
MAPH02= 170212	337#	401											
MAPH03= 170216	339#	403											
MAPH04= 170222	341#	405											
MAPH05= 170226	343#	407											
MAPH06= 170232	345#	409											
MAPH07= 170236	347#	411											
MAPH1 = 170206	399#												
MAPH10= 170242	349#												
MAPH11= 170246	351#												
MAPH12= 170252	353#												
MAPH13= 170256	355#												
MAPH14= 170262	357#												
MAPH15= 170266	359#												
MAPH16= 170272	361#												
MAPH17= 170276	363#	2334	2647										
MAPH2 = 170212	401#												
MAPH20= 170302	365#	2394	2507*	2508	2727								
MAPH21= 170306	367#												
MAPH22= 170312	369#												
MAPH23= 170316	371#												
MAPH24= 170320	373#												
MAPH25= 170326	375#												
MAPH26= 170332	377#												
MAPH27= 170336	379#												
MAPH3 = 170216	403#												
MAPH30= 170342	381#												
MAPH31= 170346	383#												
MAPH32= 170352	385#												
MAPH33= 170356	387#												
MAPH34= 170362	389#												
MAPH35= 170366	391#	4682											
MAPH36= 170372	393#												
MAPH37= 170376	395#	1725	2227	2402	2546	2560	2736						
MAPH4 = 170222	405#												
MAPH5 = 170226	407#												
MAPH6 = 170232	409#												
MAPH7 = 170236	411#												
MAPL0 = 170200	396#	1723	2222	2292	2544	2557	2934*	2943	3139				
MAPL00= 170200	332#	396	2442*	2443	2594	4586							
MAPL01= 170204	334#	398											
MAPL02= 170210	336#	400											
MAPL03= 170214	338#	402											

[illegible]

NXTTST	001324	556#	2004	2219*	2251*	2289*	2324*	2358*	2392*	2435*	2479*	2541*	2587*	2631*
		2676*	2720*	2770*	2829*	2968*	3030*	3091*	3129*	3227*	3279*	3349*	3386*	3568*
		3646*	3702*	3760*	3831*	3935*	3973*	4159*	4235*	4294*	4355*	4428*	4488*	4527*
		4577*												
OLDPC	001304	548#	1747*	1766	1780*	1794	1807*	1821	1833*	1851	1864*	1878	1893*	1907
		1934*	1939	1950	1978*	1985	2002*	2004*	2009	2032*	2041			
OLDPS	001306	549#	1748*	1765	1935*	1949	1979*	2008	2033*	2040				
OLDPSW	001310	550#	1690*	1707	1708*									
PADRSR	001222	516#	1251*	1273*										
PADRSL	001220	514#	1252*	1253	1272*	1274								
PAFLAG	005766	1104*	1969#	2007*	2195*									
PATAND	001234	522#	992*	1844*	1867*	1896*	2860*	5726	5733					
PATTOR	001236	523#	988*	1842*	1865*	1894*	2856*	5726	5733					
PCONTR	001276	545#	1983*	3681*	3738*	3809*	3880*	4270*	4330*	4404*	4477*	5707	5758	
PCPUER	001266	541#	1749*	1750	1751	1938*	1942	1947	3037*	3040	3357*	3366	3582*	3590
		3594	3941*	3950	4171*	4179	4183	5703	5705	5750				
PHIADR	001272	543#	1981*	3679*	3736*	3807*	3878*	4268*	4328*	4402*	4475*			
PIRQ =	177772	38#												
PIRQVE=	000240	148#												
PLOADR	001270	542#	1980*	1992	3678*	3735*	3806*	3877*	4267*	4327*	4401*	4474*	5707	5758
PMAINT	001300	546#	1984*	3682*	3739*	3810*	3881*	4271*	4331*	4405*	4478*	5707	5758	
PMMRO	001312	551#	2034*	5712										
PMMR1	001314	552#	2035*	5712										
PMMR2	001316	553#	2036*	5712										
PPARER	001274	544#	1982*	1986	2006	3650*	3668	3680*	3705*	3725	3737*	3767*	3795	3808*
		3838*	3866	3879*	4239*	4257	4269*	4297*	4317	4329*	4362*	4390	4403*	4435*
		4463	4476*	5707	5758									
PRO	= 000000	70#												
PR1	= 000040	71#												
PR2	= 000100	72#												
PR3	= 000140	73#												
PR4	= 000200	74#												
PR5	= 000240	75#												
PR6	= 000300	76#												
PR7	= 000340	77#												
PS	= 177776	35#	36	2052*	4571	4748								
PSW	= 177776	36#												
PWRMSG	004704	1616	1623#											
PWRVEC=	000024	142#	1589*	1590*	1598*	1613*	1614*	2064*	2065*					
RESREG=	104414	1260	1580#	1658										
RESVEC=	0000:0	137#	2073*	2080*	2107*									
RETRY	001322	555#	983*	1999	2001*	2851*								
RSIZE	001320	554#	3606*	3618	4195*	4207	5756							
SAVEA	040006	5845#												
SAVREG=	104412	1237	1579#	1643										
SDPAR0=	172260	269#												
SDPAR1=	172262	270#												
SDPAR2=	172264	271#												
SDPAR3=	172266	272#												
SDPAR4=	172270	273#												
SDPAR5=	172272	274#												
SDPAR6=	172274	275#												
SDPAR7=	172276	276#												
SDPDRO=	172220	247#												
SDPDR1=	172222	248#												
SDPDR2=	172224	249#												

[illegible]

[illegible]

[illegible]

[illegible]

2207#	2209	2235#	2237	2275#	2277	2310#	2312	2344#	2346	2378#	2380	2420#
2422	2464#	2466	2522#	2524	2573#	2575	2617#	2619	2662#	2664	2706#	2708
2750#	2752	2808#	2810	2912#	2914	2955#	2957	3011#	3013	3075#	3077	3114#
3116	3210#	3212	3268#	3270	3338#	3340	3373#	3375	3556#	3558	3630#	3632
3686#	3688	3743#	3745	3814#	3816	3917#	3919	3957#	3959	4147#	4149	4219#
4221	4275#	4277	4335#	4337	4409#	4411	4482#	4484	4521#	4523	4562#	4564
1436*	1465*	1478#										
1645	1670#											
1431*	1435*	1440	1443*	1454*	1480#							
994	1012	1020	1030	1038#								
471#	1026	1042	3182	4709*	4710*	4721	4761					
1618#												
1589#	1613	2064										
1616#												
1598	1603#											
511#	1107											
1579												
1579												
1579												
1579												
494#												
496#	1061*	4497*	4504*	4511*	4632*	4667*	5710	5719	5729	5736	5745	5761
5765	5767	5769	5771	5773	5775	5778	5780	5782				
497#	1062*	4498*	4505*	4512*	4633*	4668*	4669*	5721	5731	5738	5752	5754
5761	5765	5767	5773	5778								
498#	1063*	4551*	4634*	4670*	5719	5752	5754	5763	5765	5767	5775	5778
5782												
499#	1064*	5721	5754	5775	5780	5782						
500#	1065*	5736	5775									
501#	1066*											
1318#	1580											
2070	2072*	2077*	4737	4756#								
1302#	1579											
1597*	1603	1604*	1605*	1622#								
982#	2058											
423#	1084	2058	2060	2062	2064	2066	2067	2068	2070	2082	2086	4707
423#												
1002	1033#											
455#	460											
1#	11	16	17	18	19	20	21	22	23	508	509	510
974	975	976	977	978	993	1005	1007	1008	1013	1014	1015	1022
1023	1024	1035	1038	1041	1049	1050	1051	1052	1053	1054	1070	1073
1080	1084	1090	1107	2067	2068	2070	2082	2083	2221	2253	2291	2326
2360	2394	2437	2481	2543	2589	2633	2678	2722	2772	2831	2929	2970
3032	3093	3131	3229	3281	3351	3388	3570	3648	3704	3762	3833	3935
3975	4161	4237	4296	4357	4428	4488	4527	45				

\$TMP2	001174	504#	4553*	5763										
\$TMP3	001176	505#												
\$TMP4	001200	506#	3649*	3668	3704*	3725	3766*	3795	3837*	3866	4238*	4257	4296*	4317
		4361*	4390	4434*	4463	5758								
\$TMP5	001202	507#	2833*	2881	3652*	3673	3707*	3730	3765*	3801	3836*	3872	4241*	4262
		4299*	4322	4360*	4396	4433*	4469							
\$TN	= 000050	1#	11	2207	2218	2221#	2232	2235	2250	2253#	2263	2275	2288	2291#
		2305	2310	2323	2326#	2339	2344	2357	2360#	2373	2378	2391	2394#	2407
		2420	2434	2437#	2464	2478	2481#	2522	2540	2543#	2565	2573	2586	2589#
		2612	2617	2630	2633#	2656	2662	2675	2678#	2701	2706	2719	2722#	2745
		2750	2769	2772#	2808	2828	2831#	2912	2923	2929#	2955	2967	2970#	3011
		3029	3032#	3075	3090	3093#	3114	3128	3131#	3132	3183	3210	3226	3229#
		3262	3268	3278	3281#	3299	3338	3348	3351#	3373	3385	3388#	3556	3567
		3570#	3630	3645	3648#	3675	3686	3701	3704#	3732	3743	3759	3762#	3764
		3803	3814	3830	3833#	3917	3929	3935#	3957	3972	3975#	4147	4158	4161#
		4219	4234	4237#	4264	4275	4293	4296#	4324	4335	4354	4357#	4359	4398
		4409	4428#	4432	4471	4482	4488#	4521	4527#	4562	4569#			
\$TPB	001144	489#	1392*	1403										
\$TPFLG	001151	493#	1352	1403										
\$TPS	001142	488#	1390	1403										
\$TRAP	004506	1558#	2062											
\$TRP	= 000022	1565#	1575#	1576#	1577#	1578#	1579#	1580#	1581#	1582#	1583#			
\$TRPAD	004526	1562	1573#											
\$TSTNM	001102	472#	973	1011	1033*	1038	1042	1059	1069	1107	2927*	2928	3136*	3137
		3933*	3934	4707*										
\$TYPBN=	***** U	1579												
\$TYPDS	004262	1494#	1578											
\$TYPE	003620	1352#	1565	1574										
\$TYPEC	003764	1371	1378	1385	1390#	1391								
\$TYPEX	004032	1396	1398	1401#										
\$TYPOC	004060	1434#	1575											
\$TYPON	004074	1433	1436#	1577										
\$TYPOS	004034	1429#	1576											
\$XTSTR	002266	996#												
\$GET4=	000001	4735#	4740											
\$STRP	= 000002	1564#	1575	1576	1577	1578	1579	1580	1581	1582	1583			
\$OF ILL	004257	1430*	1434*	1444	1479#									
.	= 042050	426#	430	432#	455	456#	458#	460#	468#	514	1041	1042	1107	1224#
		1403	1548#	1600	1621	1670#	2049#	2056	2082	2083	2091#	4546	4720#	4761
		4763#	5845#	5847	5850#	5852#								

AND	1#	1756	1782	1786	1809	1813	1835	1839	1843	1866	1870	1895	1899	3317	3321
COMMEN	1#	415#													
COUNT1	1#	2590	2634	2679	2723										
DONE	1#	4703													
ENDCOM	1#	415#													
ENTRY	1#	2923	3132	3929											
ERROR	33#	1753	1761	1763	1791	1793	1818	1820	1848	1850	1875	1877	1904	1906	1944
	1946	1989	1998	2037	2233	2264	2306	2340	2374	2408	2446	2460	2498	2515	2566
	2613	2657	2702	2746	2783	2789	2796	2804	2850	2879	2945	2982	2992	2999	3006
	3045	3069	3109	3156	3178	3188	3254	3263	3328	3330	3334	3368	3402	3417	3432
	3447	3462	3477	3492	3507	3522	3537	3552	3601	3625	3670	3727	3798	3869	3952
	3991	4006	4021	4036	4051	4066	4081	4096	4111	4126	4141	4190	4214	4259	4319
	4393	4466	4499	4506	4513	4554	4635	4671							
ESCAPE	1#	415#													
MAPADD	1#	3392	3407	3422	3437	3452	3467	3482	3497	3512	3527	3542	3981	3996	4011
	4026	4041	4056	4071	4086	4101	4116	4131							
MSG1	2207#	2209													
MSG10	2464#	2466													
MSG11	2522#	2524													
MSG12	2573#	2575													
MSG13	2617#	2619													
MSG14	2662#	2664													
MSG15	2706#	2708													
MSG16	2750#	2752													
MSG17	2808#	2810													
MSG2	2235#	2237													
MSG20	2912#	2914													
MSG21	2955#	2957													
MSG22	3011#	3013													
MSG23	3075#	3077													
MSG24	3114#	3116													
MSG25	3210#	3212													
MSG26	3268#	3270													
MSG27	3338#	3340													
MSG3	2275#	2277													
MSG30	3373#	3375													
MSG31	3556#	3558													
MSG32	3630#	3632													
MSG33	3686#	3688													
MSG34	3743#	3745													
MSG35	3814#	3816													
MSG36	3917#	3919													
MSG37	3957#	3959													
MSG4	2310#	2312													
MSG40	4146#	4149													
MSG41	4219#	4221													
MSG42	4275#	4277													
MSG43	4335#	4337													
MSG44	4409#	4411													
MSG45	4481#	4484													
MSG46	4520#	4523													
MSG47	4562#	4564													
MSG5	2344#	2346													
MSG6	2378#	2380													
MSG7	2420#	2422													
MULT	1#	415#													

[illegible]

.\$CATC	1#	423
.\$CMTA	1#	461
.\$DB2D	1#	
.\$DB20	1#	1631
.\$DIV	1#	
.\$EOP	1#	4690
.\$ERRO	1#	1042
.\$ERRT	1#	
.\$MULT	1#	
.\$POWE	1#	1584
.\$RAND	1#	
.\$RDDE	1#	
.\$RDOC	1#	
.\$READ	1#	
.\$SAVE	1#	1284
.\$SB2D	1#	
.\$SB20	1#	
.\$SCOP	1#	967
.\$SIZE	1#	
.\$SUPR	1#	
.\$TRAP	1#	1549
.\$TYPB	1#	
.\$TYPD	1#	1481
.\$TYPE	1#	1330
.\$TYPO	1#	1403
.\$1170	1#	25

. ABS. 042050 000

ERRORS DETECTED: 0

DSKZ:CEKBFC.BIN,DSKZ:CEKBFC.LST/CRF/SOL=CEKBFC.SML,CEKBFC.P11
RUN-TIME: 50 73 5 SECONDS
RUN-TIME RATIO: 785/131=5.9
CORE USED: 34K (67 PAGES)