

KD11-Z

11/44 UBI MAP
CKKUABO

AH-F629B-MC
FICHE 1 OF 1

FEB 1981
COPYRIGHT © 79-80
MADE IN USA



.REM @

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

IDENTIFICATION

PRODUCT CODE:	AC-F627B-MC
PRODUCT NAME:	CKKUABO 11/44 UBI MAP
DATE CREATED:	OCTOBER, 1980
MAINTAINER:	DIAGNOSTIC ENGINEERING
AUTHOR:	JOHN CIUKAJ

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

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

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

COPYRIGHT (C) 1979, 1980 BY DIGITAL EQUIPMENT CORPORATION

39
40
41
42
43
44
45
46

HISTORY SECTION

CKKUAAO WAS RELEASED OCTOBER 1979
CKKUABO WAS RELEASED OCTOBER 1980

47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103

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 RUNNING UNDER APT
- 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-11/44
- 9) PROGRAM DESCRIPTION
- 1. ABSTRACT

THIS PROGRAM IS DESIGNED TO BE RUN ON A PDP11/44 ON WHICH THE CPU, CACHE(IF APPLICABLE), 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

104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125

INTERACTION BETWEEN MODULES PROHIBITS CLOSE ISOLATION, BUT THE FAILING FUNCTION WILL BE CALLED OUT SO THE FIELD SERVICE ENGINEER CAN COMPLETE THE ISOLATION PROCESS.

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

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

127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147

2. REQUIREMENTS

2.1 EQUIPMENT

THE BASIC PDP-11/44 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(IF APPLICABLE), 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.

149
150
151
152
153
154
155
156

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.

158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203

4. STARTING PROCEDURE

4.1 STARTING ADDRESS

PROGRAM STARTS AT ADDRESS 200

4.2 PROGRAM AND/OR OPERATOR ACTION

THE PROGRAM WILL IDENTIFY ITSELF. IT WILL THEN ASK IF YOU WANT THE EOP MESSAGES PRINTED. NO USER RESPONSE IS REQUIRED, AS A WAIT LOOP, ENDING AFTER 2 SECONDS, TERMINATES THE INPUT, AND THE DIAGNOSTIC ASSUMES YOU WANT ALL EOP MESSAGES PRINTED. IF YOU DO RESPOND WITH AN 'N', NO EOP MESSAGE FOR A PARTICULAR PASS WILL BE PRINTED AS LONG AS THERE ARE NO ERROR(S) IN THAT PASS. IF AN ERROR DOES OCCUR IN THE PARTICULAR PASS, AN EOP MESSAGE WILL PRINT SO YOU CAN IDENTIFY WHERE IN THE PASS COUNT THE ERROR OCCURED. IF AT ANY TIME YOU WISH TO HAVE A PROGRESS REPORT, TYPE ANY CHARACTER ON THE TERMINAL, AND AT THE END OF THE PASS IT IS IN, AN EOP MESSAGE WILL PRINT FOR THAT PASS ONLY. IT WILL RESUME NOT PRINTING EOP MESSAGES AS LONG AS A CHARACTER HAS NOT BEEN ENTERED AT THE TERMINAL. IF YOU ELECT TO HAVE EOP MESSAGES PRINTED, (ENTERING OTHER THAN AN 'N' OR DO NOT RESPOND IN THE 2 SECOND TIME LIMIT) THEY WILL BE AT THE END OF EACH PASS, AND 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
BIT02 -FORCE MISS ON READ,WHERE ADDRESS BIT 12 IS 0
BIT03 -FORCE MISS ON READ,WHERE ADDRESS BIT 12 IS 1
BIT09 -UNCONDITIONAL CACHE BYPASS

205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257

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<05:00>
SW07	1=	INHIBIT MULTIPLE ERROR TYPE OUTS
SW06	1=	SELECT CACHE TESTS. THIS IS USED FOR MFG. QUICK VERIFY STATION AND CAN BE SELECTED BY APT SCRIPTING. THESE TESTS ASSUME THAT ALL MODULES EXCEPT UBI MODULE ARE KNOWN GOOD.

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

SUBROUTINE TO TURN OFF AND SAVE T-BIT
 SUBROUTINE TO RESTORE T-BIT TO ITS PREVIOUS CONDITION
 SUBROUTINE TO CLEAR ALL OF THE MAP REGISTERS
 SUBROUTINE TO EXTRACT MAP ADDRESS FROM PAR CONTENTS

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

258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292

5.3 RUNNING UNDER APT

THE EXECUTION TIMES PROVIDED IN THE APT SCRIPT THAT FOLLOWS ARE FOR EXECUTION WITH A 11/44 PROCESSOR, CACHE, 16K CORE MEMORY, AND 300 BAUD. THE FOLLOWING IS A PROGRAM LOAD FILE USED BY APT:

1. E TABLE 'A' IS USED FOR APT DUMP MODE.
 - A. IN ADDITION TO NORMAL CPU DIAGNOSTIC TESTS THIS TABLE WILL SELECT THE OPTIONAL CACHE TESTS, (\$SWREG=100) AND INHIBIT ITERATIONS(\$SWREG=4000)
2. E TABLE 'B' IS USED FOR APT QV MODE WHILE RUNNING ON A MANUFACTURING QV STATION. IT ACCOMPLISHES WHAT ETABLE 'A' DOES BUT ADDITIONALLY SUPPRESSES TYPEOUTS.(\$ENVM=240)
3. ETABLE 'C' IS USED FOR APT QV OR RUNTIME MODES WHILE RUNNING ON SYSTEMS OTHER THAN MFG. QV STATIONS. THIS TABLE DESELECTS THE OPTIONAL CACHE TESTS.

	1ST PASS RUN TIME	LONGEST TEST TIME	ADDITIONAL RUN TIME	
	10	5	0	
.....		E TABLES	
		A	B	C
E-MODE/S-MODE (\$ENVM/\$ENV)		200/000	240/001	240/001
SWITCH REGISTER 1 (\$SWREG)		004100	0004100	004000
SWITCH REGISTER 2		000000	000000	000000
CPU TYPE/OPTIONS		00/0000	00/0000	00/0000

293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342

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 SW09=1, 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 SW09=0, 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: AN EXAMPLE:

THE FOLLOWING REGISTERS TIMED OUT WHEN REFERENCED

REG.ADR	TESTNO	ERRORPC
170210	000001	015226
170212	000001	015232
.	.	.
.	TO	.
.	.	.
170372	000001	015232
170374	000001	015232
170376	000001	015232

SUMMARY OF MAP REGISTERS THAT TIMED OUT ON READ

REGADRS	REGADRS	#ERRORS	TESTNO	ERRORPC
'OR'	'AND'			
170376	170210	32	000001	010530

343
344
345
346
347
348
349
350
351
352

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

353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408

8. MISCELLANEOUS

8.1 EXECUTION TIME

THE RUN TIME FOR ANY PASS IS APPROXIMATELY 3 SECONDS.

8.2 ADDRESS GENERATION IN THE PDP-11/44

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. P.A.R. PAGE NUMBER (0-7)	15 14 13
A2. OFFSET (FROM VIRTUAL ADDRESS)	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. PHYS ADDRS (A2+B)	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. PHYS ADDR (C3+D)	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.

409
410
411
412
413
414

9. PROGRAM DESCRIPTION

THE ASSEMBLED LISTING,CKKUABO.SEQ, 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.ⓐ

558

```
.TITLE CKKUABO 11/44 UBI MAP
:*COPYRIGHT (C) OCTOBER 1980
:*DIGITAL EQUIPMENT CORP.
:*MAYNARD, MASS. 01754
:*
:*PROGRAM BY DAN P. MILLEVILLE
:*
:*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
:*PACKAGE (MAINDEC-11-DZQAC-A6).
:*
```

559

.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<4:0>
7	INHIBIT MULTIPLE ERROR TYPEOUTS
6	SELECT CACHE-CIS TESTS
5	SELECT MEMORY ON UNIBUS TEST

560

561

562

```
.SBTTL BASIC DEFINITIONS
;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
001100 STACK= 1100 ;;FIRST ADDRESS OF THE STACK
001100 KERSTK= STACK ;;KERNEL STACK
000700 SUPSTK= STACK-200 ;;SUPERVISOR STACK
000600 USESTK= STACK-300 ;;USER STACK
104000 ERROR=EMT
000004 SCOPE=IOT
177776 PS= 177776 ;;PROCESSOR STATUS WORD
177776 PSW=PS
177774 STKLMT= 177774 ;;STACK LIMIT REGISTER
177772 PIRQ= 177772 ;;PROGRAM INTERRUPT REQUEST REGISTER
;SWR= 177570 ;;SWITCH REGISTER (ELIMINATED FOR CKKUAB)
;DISPLAY=SWR ;;(ELIMINATED FOR CKKUAB)
;*MISCELLANEOUS DEFINITIONS
000011 HT= 11 ;;CODE FOR HORIZONTAL TAB
000012 LF= 12 ;;CODE LINE FEED
000015 CR= 15 ;;CODE CARRIAGE RETURN
000200 CRLF= 200 ;;CODE FOR CARRIAGE RETURN-LINE FEED
;*GENERAL PURPOSE REGISTER DEFINITIONS
000000 R0= %0 ;;GENERAL REGISTER
000001 R1= %1 ;;GENERAL REGISTER
000002 R2= %2 ;;GENERAL REGISTER
000003 R3= %3 ;;GENERAL REGISTER
000004 R4= %4 ;;GENERAL REGISTER
000005 R5= %5 ;;GENERAL REGISTER
000006 R6= %6 ;;GENERAL REGISTER
000007 R7= %7 ;;GENERAL REGISTER
000000 R10=R0
000001 R11=R1
000002 R12=R2
000003 R13=R3
000004 R14=R4
000005 R15=R5
000006 SP=R6
000006 KSP=SP
000006 SSP=SP
000006 USP=SP
000007 PC=R7
;*PRIORITY LEVEL DEFINITIONS
000000 PR0= 0 ;;PRIORITY LEVEL 0
000040 PR1= 40 ;;PRIORITY LEVEL 1
000100 PR2= 100 ;;PRIORITY LEVEL 2
000140 PR3= 140 ;;PRIORITY LEVEL 3
000200 PR4= 200 ;;PRIORITY LEVEL 4
000240 PR5= 240 ;;PRIORITY LEVEL 5
000300 PR6= 300 ;;PRIORITY LEVEL 6
000340 PR7= 340 ;;PRIORITY LEVEL 7
;*SWITCH REGISTER SWITCH DEFINITIONS
100000 SW15= 100000
040000 SW14= 40000
```

020000
010000
004000
002000
001000
000400

SW13= 20000
SW12= 10000
SW11= 4000
SW10= 2000
SW09= 1000
SW08= 400

E 2

SEQ 0017

C
U

```
000200 SW07= 200
000100 SW06= 100
000040 SW05= 40
000020 SW04= 20
000010 SW03= 10
000004 SW02= 4
000002 SW01= 2
000001 SW00= 1
001000 SW9=SW09
000400 SW8=SW08
000200 SW7=SW07
000100 SW6=SW06
000040 SW5=SW05
000020 SW4=SW04
000010 SW3=SW03
000004 SW2=SW02
000002 SW1=SW01
000001 SW0=SW00
;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
100000 BIT15= 100000
040000 BIT14= 40000
020000 BIT13= 20000
010000 BIT12= 10000
004000 BIT11= 4000
002000 BIT10= 2000
001000 BIT09= 1000
000400 BIT08= 400
000200 BIT07= 200
000100 BIT06= 100
000040 BIT05= 40
000020 BIT04= 20
000010 BIT03= 10
000004 BIT02= 4
000002 BIT01= 2
000001 BIT00= 1
001000 BIT9=BIT09
000400 BIT8=BIT08
000200 BIT7=BIT07
000100 BIT6=BIT06
000040 BIT5=BIT05
000020 BIT4=BIT04
000010 BIT3=BIT03
000004 BIT2=BIT02
000002 BIT1=BIT01
000001 BIT0=BIT00
;*BASIC "CPU" TRAP VECTOR ADDRESSES
000004 ERRVEC= 4 ;;TIME OUT AND OTHER ERRORS
000010 RESVEC= 10 ;;RESERVED AND ILLEGAL INSTRUCTIONS
000014 TBITVEC=14 ;;"T" BIT
000014 TRTVEC= 14 ;;TRACE TRAP
000014 BPTVEC= 14 ;;BREAKPOINT TRAP (BPT)
```

000020
000024
000030
000034
000060
000064

IOTVEC= 20
PWRVEC= 24
EMTVEC= 30
TRAPVEC= 34
TKVEC= 60
TPVEC= 64

G 2
:: INPUT/OUTPUT TRAP (IOT) **SCOPE**
:: POWER FAIL
:: EMULATOR TRAP (EMT) **ERROR**
:: "TRAP" TRAP
:: TTY KEYBOARD VECTOR
:: TTY PRINTER VECTOR

SEQ 0019

C
S

BASIC DEFINITIONS
000114
000240
000250

CACHVEC=114
PIRQVEC=240
MMVEC= 250

::CACHE ERROR INTERRUPT VECTOR
::PROGRAM INTERRUPT REQUEST VECTOR
::MEMORY MANAGEMENT VECTOR

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


```
.SBTTL MEMORY MANAGEMENT DEFINITIONS
;*MEMORY MANAGEMENT STATUS REGISTER ADDRESSES
177572 MMR0= 177572
177574 MMR1= 177574
177576 MMR2= 177576
172516 MMR3= 172516
177572 SR0=MMR0
177574 SR1=MMR1
177576 SR2=MMR2
172516 SR3=MMR3
;*USER 'I' PAGE DESCRIPTOR REGISTERS
177600 UIPDR0= 177600
177602 UIPDR1= 177602
177604 UIPDR2= 177604
177606 UIPDR3= 177606
177610 UIPDR4= 177610
177612 UIPDR5= 177612
177614 UIPDR6= 177614
177616 UIPDR7= 177616
;*USER 'D' PAGE DESCRIPTOR REGISTORS
177620 UDPDR0= 177620
177622 UDPDR1= 177622
177624 UDPDR2= 177624
177626 UDPDR3= 177626
177630 UDPDR4= 177630
177632 UDPDR5= 177632
177634 UDPDR6= 177634
177636 UDPDR7= 177636
;*USER 'I' PAGE ADDRESS REGISTERS
177640 UIPAR0= 177640
177642 UIPAR1= 177642
177644 UIPAR2= 177644
177646 UIPAR3= 177646
177650 UIPAR4= 177650
177652 UIPAR5= 177652
177654 UIPAR6= 177654
177656 UIPAR7= 177656
;*USER 'D' PAGE ADDRESS REGISTERS
177660 UDPAR0= 177660
177662 UDPAR1= 177662
177664 UDPAR2= 177664
177666 UDPAR3= 177666
177670 UDPAR4= 177670
177672 UDPAR5= 177672
177674 UDPAR6= 177674
177676 UDPAR7= 177676
;*SUPERVISOR 'I' PAGE DESCRIPTOR REGISTERS
172200 SIPDR0= 172200
172202 SIPDR1= 172202
172204 SIPDR2= 172204
172206 SIPDR3= 172206
```


172210
172212
172214
172216

172220

SIPDR4= 172210
SIPDR5= 172212
SIPDR6= 172214
SIPDR7= 172216
;*SUPERVISOR 'D' PAGE DESCRIPTOR REGISTERS
SDPDRO= 172220

```
172222 SDPDR1= 172222
172224 SDPDR2= 172224
172226 SDPDR3= 172226
172230 SDPDR4= 172230
172232 SDPDR5= 172232
172234 SDPDR6= 172234
172236 SDPDR7= 172236
;*SUPERVISOR 'I' PAGE ADDRESS REGISTERS
172240 SIPAR0= 172240
172242 SIPAR1= 172242
172244 SIPAR2= 172244
172246 SIPAR3= 172246
172250 SIPAR4= 172250
172252 SIPAR5= 172252
172254 SIPAR6= 172254
172256 SIPAR7= 172256
;*SUPERVISOR 'D' PAGE ADDRESS REGISTERS
172260 SDPAR0= 172260
172262 SDPAR1= 172262
172264 SDPAR2= 172264
172266 SDPAR3= 172266
172270 SDPAR4= 172270
172272 SDPAR5= 172272
172274 SDPAR6= 172274
172276 SDPAR7= 172276
;*KERNEL 'I' PAGE DESCRIPTOR REGISTERS
172300 KIPDR0= 172300
172302 KIPDR1= 172302
172304 KIPDR2= 172304
172306 KIPDR3= 172306
172310 KIPDR4= 172310
172312 KIPDR5= 172312
172314 KIPDR6= 172314
172316 KIPDR7= 172316
;*KERNEL 'D' PAGE DESCRIPTOR REGISTERS
172320 KDPDR0= 172320
172322 KDPDR1= 172322
172324 KDPDR2= 172324
172326 KDPDR3= 172326
172330 KDPDR4= 172330
172332 KDPDR5= 172332
172334 KDPDR6= 172334
172336 KDPDR7= 172336
;*KERNEL 'I' PAGE ADDRESS REGISTERS
172340 KIPAR0= 172340
172342 KIPAR1= 172342
172344 KIPAR2= 172344
172346 KIPAR3= 172346
172350 KIPAR4= 172350
172352 KIPAR5= 172352
172354 KIPAR6= 172354
```

172356

172360

172362

172364

172366

KIPAR7= 172356
;*KERNEL 'D' PAGE ADDRESS REGISTERS
KDPAR0= 172360
KDPAR1= 172362
KDPAR2= 172364
KDPAR3= 172366

N 2

SEQ 0026

172370	KDPAR4= 172370
172372	KDPAR5= 172372
172374	KDPAR6= 172374
172376	KDPAR7= 172376

```
.SBTTL UNIBUS MAP REGISTER DEFINITIONS
;*THE LOWER 16 BITS OF THE MAP REGISTERS ARE LABELED 'MAPLXX'
;*THE UPPER 6 BITS OF THE MAP REGISTERS ARE LABELED 'MAPHXX'
170200 MAPL00 = 170200
170202 MAPH00 = 170202
170204 MAPL01 = 170204
170206 MAPH01 = 170206
170210 MAPL02 = 170210
170212 MAPH02 = 170212
170214 MAPL03 = 170214
170216 MAPH03 = 170216
170220 MAPL04 = 170220
170222 MAPH04 = 170222
170224 MAPL05 = 170224
170226 MAPH05 = 170226
170230 MAPL06 = 170230
170232 MAPH06 = 170232
170234 MAPL07 = 170234
170236 MAPH07 = 170236
170240 MAPL10 = 170240
170242 MAPH10 = 170242
170244 MAPL11 = 170244
170246 MAPH11 = 170246
170250 MAPL12 = 170250
170252 MAPH12 = 170252
170254 MAPL13 = 170254
170256 MAPH13 = 170256
170260 MAPL14 = 170260
170262 MAPH14 = 170262
170264 MAPL15 = 170264
170266 MAPH15 = 170266
170270 MAPL16 = 170270
170272 MAPH16 = 170272
170274 MAPL17 = 170274
170276 MAPH17 = 170276
170300 MAPL20 = 170300
170302 MAPH20 = 170302
170304 MAPL21 = 170304
170306 MAPH21 = 170306
170310 MAPL22 = 170310
170312 MAPH22 = 170312
170314 MAPL23 = 170314
170316 MAPH23 = 170316
170320 MAPL24 = 170320
170320 MAPH24 = 170320
170324 MAPL25 = 170324
170326 MAPH25 = 170326
170330 MAPL26 = 170330
170332 MAPH26 = 170332
170334 MAPL27 = 170334
170336 MAPH27 = 170336
```

170340
170342
170344
170346
170350
170352

MAPL30 = 170340
MAPH30 = 170342
MAPL31 = 170344
MAPH31 = 170346
MAPL32 = 170350
MAPH32 = 170352

D 3

SEQ 0029

CI
EF

170354	MAPL33 = 170354
170356	MAPH33 = 170356
170360	MAPL34 = 170360
170362	MAPH34 = 170362
170364	MAPL35 = 170364
170366	MAPH35 = 170366
170370	MAPL36 = 170370
170372	MAPH36 = 170372
170374	MAPL37 = 170374
170376	MAPH37 = 170376
170200	MAPL0=MAPL00
170202	MAPH0=MAPH00
170204	MAPL1=MAPL01
170206	MAPH1=MAPH01
170210	MAPL2=MAPL02
170212	MAPH2=MAPH02
170214	MAPL3=MAPL03
170216	MAPH3=MAPH03
170220	MAPL4=MAPL04
170222	MAPH4=MAPH04
170224	MAPL5=MAPL05
170226	MAPH5=MAPH05
170230	MAPL6=MAPL06
170232	MAPH6=MAPH06
170234	MAPL7=MAPL07
170236	MAPH7=MAPH07

.....

566

000000

.SBTTL TRAP CATCHER

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


```
000200 000200 010000 .SBTTL STARTING ADDRESS(ES)  
000137 .=200  
JMP @#START ;;JUMP TO STARTING ADDRESS OF PROGRAM
```

568

```

.SBTTL          ACT11 HOOKS
:*****
:*THE FOLLOWING LOCATIONS ARE SETUP TO BE USED WITH ACT11
:*
:*LOCATION 46 WILL CONTAIN THE ADDRESS OF THE LOGICAL
:*END OF THE PROGRAM.
:*LOCATION 52 IS USED TO SPECIFY PROGRAM OPERATING REQUIREMENTS
:*AND/OR RESTRICTIONS. THIS IS ACCOMPLISHED BY SETTING VARIOUS BITS
:*TO A ONE OR A ZERO. THE BITS USED AND THERE MEANING ARE:
:*
:*      BIT 15=1 PROGRAM SHOULD BE POWER FAILED WHILE RUNNING
:*              =0 NO POWER FAIL DESIRED
:*
:*      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

```

```

000204
000046
000046 021534
000052 000052
000052 000000
000204

```

570

.SBTTL COMMON TAGS

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

Address	Value	Label	Format	Value	Description
001100	001100	\$CMTAG:			:: START OF COMMON TAGS
001100	000	\$PASS:	.WORD	0	:: && CONTAINS PASS COUNT (ELIMINATED FOR CKKUAB)
001101	000	\$TSTNM:	.BYTE	0	:: CONTAINS THE TEST NUMBER
001102	000000	\$ERFLG:	.BYTE	0	:: CONTAINS ERROR FLAG
001104	000000	\$ICNT:	.WORD	0	:: CONTAINS SUBTEST ITERATION COUNT
001106	000000	\$LPADR:	.WORD	0	:: CONTAINS SCOPE LOOP
001110	000000	\$LPERR:	.WORD	0	:: CONTAINS SCOPE RETURN FOR ERRORS
001112	000	\$ERTTL:	.WORD	0	:: CONTAINS TOTAL ERRORS DETECTED
001113	001	\$ITEMB:	.BYTE	0	:: CONTAINS ITEM CONTROL BYTE
001114	000000	\$ERMAX:	.BYTE	1	:: CONTAINS MAX. ERRORS PER TEST
001116	000000	\$ERRPC:	.WORD	0	:: CONTAINS PC OF LAST ERROR INSTRUCTION
001120	000000	\$GDADR:	.WORD	0	:: CONTAINS OF 'GOOD' DATA
001122	000000	\$BDADR:	.WORD	0	:: CONTAINS OF 'BAD' DATA
001124	000000	\$GDDAT:	.WORD	0	:: CONTAINS 'GOOD' DATA
001126	000000	\$BDDAT:	.WORD	0	:: CONTAINS 'BAD' DATA
001134	177560	\$TKS:	.WORD	177560	:: RESERVED--NOT TO BE USED
001136	177562	\$TKB:	.WORD	177562	:: TTY KBD STATUS
001140	177564	\$TPS:	.WORD	177564	:: TTY KBD BUFFER
001142	177566	\$TPB:	.WORD	177566	:: TTY PRINTER STATUS REG.
001144	000	\$NULL:	.BYTE	0	:: TTY PRINTER BUFFER REG.
001145	002	\$FILLS:	.BYTE	2	:: CONTAINS NULL CHARACTER FOR FILLS
001146	012	\$FILLC:	.BYTE	12	:: CONTAINS # OF FILLER CHARACTERS REQUIRED
001147	000	\$TPFLG:	.BYTE	0	:: INSERT FILL CHARS. AFTER A 'LINE FEED'
001150	177570	\$WR:	.WORD	177570	:: 'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)
001152	177570	\$DISPLAY:	.WORD	177570	:: && LOCATION FOR STORING SWITCH REGISTER
001154	000000	\$REGAD:	.WORD	0	:: && DISPLAY REGISTER
001156	000006	\$REG0:	.REPT	\$CM3	:: CONTAINS THE FROM
001160	000000	\$REG1:	.WORD	0	:: WHICH (\$REG0) WAS OBTAINED
001162	000000	\$REG2:	.WORD	0	:: CONTAINS ((\$REGAD)+0)
001164	000000	\$REG3:	.WORD	0	:: CONTAINS ((\$REGAD)+2)
001166	000000	\$REG4:	.WORD	0	:: CONTAINS ((\$REGAD)+4)
001170	000000	\$REG5:	.WORD	0	:: CONTAINS ((\$REGAD)+6)
001172	000007	\$TMP0:	.REPT	7	:: CONTAINS ((\$REGAD)+10)
001174	000000	\$TMP1:	.WORD	0	:: CONTAINS ((\$REGAD)+12)
001176	000000	\$TMP2:	.WORD	0	:: USER DEFINED
001200	000000	\$TMP3:	.WORD	0	:: USER DEFINED
001202	000000	\$TMP4:	.WORD	0	:: USER DEFINED
001204	000000	\$TMP5:	.WORD	0	:: USER DEFINED
001206	000000	\$TMP6:	.WORD	0	:: USER DEFINED
001210	000000	\$TIMES:		0	:: USER DEFINED
001212	000000	\$ESCAPE:		0	:: MAX. NUMBER OF ITERATIONS
001214	207	\$BELL:	.ASCIIZ	<207><377><377>	:: ESCAPE ON ERROR

001220 077
001221 015
001222 012 000
001224 000000
001226 000000

\$QUES: .ASCII /?/
\$CRLF: .ASCII <15>
\$LF: .ASCIZ <12>
PADRSL: .WORD 0
PADRSH: .WORD 0

J 3

::QUESTION MARK
::CARRIAGE RETURN
::LINE FEED
:HOLDS THE LOWER 16 BITS OF A 22 BIT ADDRESS
:GENERATED FOR TYPE OUT.
:HOLDS THE UPPER 6 BITS OF A 22 BIT ADDRESS

SEQ 0035

CI
EF

001230	000000	000077	ADRAND: .WORD	0,77	:GENERATED FOR TYPE OUT.
001234	000000	000077	ADDROR: .WORD	0,77	:LOGICAL AND OF FAILING ADDRESSES
001240	000000	000077	DATAND: .WORD	0,77	:LOGICAL OR OF FAILING ADDRESSES
001244	000000	000077	DATAOR: .WORD	0,77	:LOGICAL AND OF BAD DATA
001250	000000		PATAND: .WORD	0	:LOGICAL OR OF BAD DATA
001252	000000		PATTOR: .WORD	0	:LOGICAL AND OF PATTERN LOADED
001254	000000		LOWEST: .WORD	0	:LOGICAL OR OF PATTERN LOADED
001256	000000		HIGEST: .WORD	0	:HOLDS NUMBER TO PUT IN PAR TO CAUSE THE
001260	000000		UBMLOW: .WORD	0	:LOWEST USABLE MAP REGISTER TO RESPOND
001262	000000		UBMHI: .WORD	0	:HOLDS NUMBER TO PUT IN PAR TO CAUSE THE
001264	000000		MMRLOW: .WORD	0	:HIGHEST USABLE MAP REGISTER TO RESPOND
001266	000000		MMRHI: .WORD	0	:HOLDS NUMBER TO PUT IN PAR TO SIGNAL 1ST
001270	000000		UBRLOW: .WORD	0	:ADDRESS OF UNIBUS MEMORY
001272	000000		UBRHI: .WORD	0	:HOLDS NUMBER TO PUT IN PAR TO SIGNAL LAST
001274	000000		BUPWIN: .WORD	0	:BLOCK OF 4K OF UNIBUS MEMORY
001276	000000		LREGL: .WORD	0	:HOLDS LOWEST MAP REGISTER NUMBER FROM 'LOWEST:'
001300	000000		LREGU: .WORD	0	:HOLDS HIGHEST MAP REGISTER NUMBER FROM 'HIGEST:'
001302	000000		FRRCNT: .WORD	0	:HOLDS LOWEST MAP REGISTER NUMBER FROM 'UBMLOW:'
001304	000000		CNTR: .WORD	0	:HOLDS HIGHEST MAP REGISTER NUMBER FROM 'UBMHI:'
001306	000000		FLAG: .WORD	0	:HOLDS LOWEST USEABLE PAR OF UPPER WINDOW
001310	000000		TESTNO: .WORD	0	:HOLDS I/O PAGE ADDR OF LOW 16 BITS OF
001312	000000		CPUEXP: .WORD	0	:THE LOWEST USABLE MAP REGISTER
001314	000000		PCPUER: .WORD	0	:HOLDS I/O PAGE ADDR OF HIGH 6 BITS OF
001316	000000		PPARER: .WORD	0	:OF THE LOWEST USABLE MAP REGISTER
001320	000000		PCONTR: .WORD	0	:MULTIPLE ERROR ERROR COUNTER
001322	000000		PMAINT: .WORD	0	:AUXILIARY COUNTER
001324	000000		BADPC: .WORD	0	:FLAG TO INDICATE TO LAST PROGRAM PASS N
001326	000000		OLDPC: .WORD	0	:HOLDS TEST NUMBER FOR ERROR TYPE OUTS
001330	000000		OLDPS: .WORD	0	:HOLDS THE EXPECTED CPU ERROR CODE
001332	000000		OLDPSW: .WORD	0	:HOLDS RECEIVED CPU ERROR CONDITION
001334	000000		PMMR0: .WORD	0	:HOLDS RECEIVED PARITY ERROR CONDITION
001336	000000		PMMR1: .WORD	0	:HOLDS CONTENTS OF CONTROL REGISTER
001340	000000		PMMR2: .WORD	0	:HOLDS CONTENTS OF MAINTENANCE REGISTER
001342	000000		RSIZE: .WORD	0	:HOLDS PC OF INST THAT CAUSED TRAP
001344	000000		RETRY: .WORD	0	:HOLDS THE RETURN ADDRESS AFTER A TRAP
001346	000000		NXTTST: .WORD	0	:HOLDS THE OLD PROCESSOR STATUS
001350	000200		DATA: .WORD	200	:HOLDS OLD PSW FOR TBITRESTORE

:PATTERN TO BE USED TO LOAD INTO MEMORY

.SBTTL ERROR POINTER TABLE

```

*****
*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
*NOTE1: IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
*NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
*      EM      ;;POINTS TO THE ERROR MESSAGE
*      DH      ;;POINTS TO THE DATA HEADER
*      DT      ;;POINTS TO THE DATA
*      DF      ;;POINTS TO THE DATA FORMAT

```

001352	\$ERRTB:			
571	:ITEM 1			
572	001352	022062	.WORD	EM1 ;NOT THE CORRECT CPU TRAP CONDITION THROUGH ERRVEC (#004)
573	001354	026052	.WORD	DH1 ;RECEIVD EXPECTD TESTNO PC AT ABORT
574	001356	030262	.WORD	DT1 ;PCPUER,CPUEXP,TESTNO,BADPC,0
575	001360	030722	.WORD	DF1 ; 0, 0, 0, 0
576				
577	:ITEM 2			
578	001362	022147	.WORD	EM2 ;UNEXPECTED CPU TRAP THROUGH ERRVEC (#004)
579	001364	026111	.WORD	DH2 ;RECEIVD TESTNO PC AT ABORT
580	001366	030274	.WORD	DT2 ;PCPUER,TESTNO,BADPC,0
581	001370	030722	.WORD	DF1 ; 0, 0, 0, 0
582				
583	:ITEM 3			
584	001372	022221	.WORD	EM3 ;MEMORY MANAGEMENT TRAP, MEMORY MANAGEMENT STATUS REGISTERS
585	001374	026140	.WORD	DH3 ;STATUS AUTOI/D VIRTADR
586				REGISTR REGISTR REGISTR TESTNO PC AT ABORT
587	001376	030304	.WORD	DT3 ;PMMR0,PMMR1,PMMR2,TESTNO,BADPC,0
588	001400	030722	.WORD	DF1 ; 0, 0, 0, 0, 0
589				
590	:ITEM 4			
591	001402	022327	.WORD	EM4 ;SUMMARY OF MAP REGISTERS THAT TIMED OUT ON READ
592	001404	026237	.WORD	DH4 ;REGADRS REGADRS
593				;'OR' 'AND' #ERRORS TESTNO ERR PC
594	001406	030320	.WORD	DT4 ;ADDROR,ADRAND,ERRCNT,TESTNO,\$ERRPC,0
595	001410	030727	.WORD	DF4 ; 2, 2, 1, 0, 0
596				
597	:ITEM 5			
598	001412	022407	.WORD	EM5 ;SUMMARY OF DUAL ADDRESSING ERRORS ON LOADING MAP REGISTERS
599	001414	026334	.WORD	DH5 ;REGLOAD REGLOAD REGDUAL REGDUAL
600				;'OR' 'AND' 'OR' 'AND' #ERRORS TESTNO
601	001416	030334	.WORD	DT5 ;ADDROR,ADRAND,DATAOR,DATAND,ERRCNT,TESTNO,0
602	001420	030734	.WORD	DF5 ; 2, 2, 2, 2, 1, 0
603				
604	:ITEM 6			
605	001422	022502	.WORD	EM6 ;SUMMARY OF BIT PATTERN FAILURES IN LOWER 16 BITS OF MAP REGISTERS
606	001424	026471	.WORD	DH6 ;MAPREG MAPREG EXPECTD EXPECTD RECEIVD RECEIVD
607				;'OR' 'AND' 'OR' 'AND' 'OR' 'AND' #ERRORS TESTNO
608	001426	030352	.WORD	DT6 ;ADDROR,ADRAND,PATTOR,PATAND,DATAOR,DATAND,ERRCNT,TESTNO,0
609	001430	030742	.WORD	DF6 ; 2, 2, 0, 0, 0, 0, 1, 0

Line	Address	Value	Field	Description
610			:ITEM 7	
611	001432	022604	.WORD EM7	:SUMMARY OF BIT PATTERN FAILURES IN UPPER 6 BITS OF MAP REGISTERS
612	001434	026471	.WORD DH6	:MAPREG MAPREG EXPECTD EXPECTD RECEIVD RECEIVD
613				: "OR" "AND" "OR" "AND" "OR" "AND" #ERRORS TESTNO
614	001436	030352	.WORD DT6	:ADDROR,ADRAND,PATTOR,PATAND,DATAOR,DATAND,ERRCNT,TESTNO,0
615	001440	030742	.WORD DF6	: 2, 2, 0, 0, 0, 0, 1, 0
616				
617			:ITEM 10	
618	001442	022705	.WORD EM10	:CAN'T GET TO MAIN MEMORY FROM UNIBUS WITH THE MAP OFF
619				:SO I'LL JUMP TO THE SIZE JUMPER TEST FOR VERIFICATION
620	001444	026660	.WORD DH10	:TESTNO ERR PC
621	001446	030374	.WORD DT10	:TESTNO,\$ERRPC,0
622	001450	030722	.WORD DF1	:0, 0
623				
624			:ITEM 11	
625	001452	023061	.WORD EM11	:SUMMARY OF COUNT PATTERN FAILURES ON THE UNIBUS DATA PATH
626	001454	026677	.WORD DH11	:EXPECTD EXPECTD RECEIVD RECEIVD
627				: "OR" "AND" "OR" "AND" #ERRORS TESTNO
628	001456	030402	.WORD DT11	:PATTOR,PATAND,DATAOR,DATAND,ERRCNT,TESTNO,0
629	001460	030752	.WORD DF11	: 0, 0, 0, 0, 1, 0
630				
631			:ITEM 12	
632	001462	023153	.WORD EM12	:UNIBUS MAP IS RELOCATING WHEN NOT ENABLED
633	001464	026660	.WORD DH10	:TESTNO ERR PC
634	001466	030374	.WORD DT10	:TESTNO,\$ERRPC,0
635	001470	030722	.WORD DF1	: 0, 0
636				
637			:ITEM 13	
638	001472	023225	.WORD EM13	:CANNOT USE ANY OF THE MAP REGISTERS OR PHYSICAL
639				:ADDRESS BIT14 IS STUCK LOW, MUST RESTART PROGRAM
640				:IF YOU DON'T LOOP ON THIS PROBLEM.
641	001474	026660	.WORD DH10	:TESTNO ERR PC
642	001476	030374	.WORD DT10	:TESTNO,\$ERRPC,0
643	001500	030722	.WORD DF1	: 0, 0
644				
645			:ITEM 14	
646	001502	023431	.WORD EM14	:SUMMARY OF UNIBUS ADDRESS ERRORS, WITH MAP RELOCATION DISABLED
647	001504	026677	.WORD DH11	:EXPECTD EXPECTD RECEIVD RECEIVD
648				: "OR" "AND" "OR" "AND" #ERRORS TESTNO
649	001506	030420	.WORD DT14	:ADDROR,ADRAND,DATAOR,DATAND,ERRCNT,TESTNO,0
650	001510	030760	.WORD DF14	: 2, 2, 0, 0, 1, 0
651				
652			:ITEM 15	
653	001512	023534	.WORD EM15	:MAIN MEMORY TIME OUT OVER THE UNIBUS DID NOT OCCUR PROPERLY.
654	001514	027034	.WORD DH15	:CONDITN CONDITN
655				:EXPECTD RECEIVD TESTNO ERR PC
656	001516	030436	.WORD DT15	:CPUEXP,PCPUER,TESTNO,\$ERRPC,0
657	001520	030722	.WORD DF1	: 0, 0, 0, 0
658				
659			:ITEM 16	
660	001522	023627	.WORD EM16	:SUMMARY OF DUAL MAPPING ERRORS
661	001524	026677	.WORD DH11	:EXPECTD EXPECTD RECEIVD RECEIVD
662				: "OR" "AND" "OR" "AND" #ERRORS TESTNO
663	001526	030420	.WORD DT14	:ADDROR,ADRAND,DATAOR,DATAND,\$ERRPC,TESTNO,0
664	001530	030734	.WORD DF5	: 2, 2, 2, 2, 1, 0

665					
666	001532	023725	:ITEM 17		
667	001534	026660	.WORD	EM17	:NO UNIBUS MEMORY EXISTS
668	001536	030374	.WORD	DH10	:TESTNO ERR PC
669	001540	030722	.WORD	DT10	:TESTNO,\$ERRPC,0
670			.WORD	DF1	: 0, 0
671			:ITEM 20		
672	001542	023755	.WORD	EM20	:INTERRUPT/ABORT LOGIC TESTS TRAP TO LOCATION 114 DID NOT OCCUR
673	001544	026660	.WORD	DH10	:TESTNO ERR PC
674	001546	030374	.WORD	DT10	:TESTNO,\$ERRPC,0
675	001550	030722	.WORD	DF1	: 0, 0
676			:ITEM 21		
677			.WORD	EM21	:INTERRUPT/ABORT TESTS R4 WAS OVERWRITTEN WITH
678	001552	024054	.WORD	DH10	:DATA INDICATING THAT INSTRUCTION WAS NOT ABORTED
679			.WORD	DT10	:TESTNO ERR PC
680	001554	026660	.WORD	DT10	:TESTNO,\$ERRPC,0
681	001556	030374	.WORD	DF1	: 0, 0
682	001560	030722	.WORD	DF1	: 0, 0
683			:ITEM 22		
684			.WORD	EM22	:INTERRUPT/ABORT TESTS TRAP DID NOT OCCUR DUE TO ABORT
685	001562	024132	.WORD	DH10	:TESTNO ERR PC
686	001564	026660	.WORD	DT10	:TESTNO,\$ERRPC,0
687	001566	030374	.WORD	DF1	: 0, 0
688	001570	030722	.WORD	DF1	: 0, 0
689			:ITEM 23		
690			.WORD	EM23	:LMA NOT LOADED PROPERLY
691	001572	024220	.WORD	DH23	:TESTNO ERR PC LMAEXP LMARCV
692	001574	027113	.WORD	DT23	:TESTNO,\$ERRPC,EADRES,EADRS2,0
693	001576	030450	.WORD	DF23	: 0, 0, 2, 2
694	001600	030766	.WORD	DF23	: 0, 0, 2, 2
695			:ITEM 24		
696			.WORD	EM24	:LMA FORCE JUMPER BIT NOT ZERO
697	001602	024250	.WORD	DH24	:TESTNO ERR PC LMAEXP LMARCV
698	001604	027154	.WORD	DT24	:TESTNO,\$ERRPC,\$REG1,LMAHI,0
699	001606	030462	.WORD	DF1	: 0, 0, 0, 0
700	001610	030722	.WORD	DF1	: 0, 0, 0, 0
701			:ITEM 25		
702			.WORD	EM25	:LMA FORCE JUMPER BIT NOT SET
703	001612	024306	.WORD	DH24	:TESTNO ERR PC LMAEXP LMARCV
704	001614	027154	.WORD	DT24	:TESTNO,\$ERRPC,\$REG1,LMAHI,0
705	001616	030462	.WORD	DF1	: 0, 0, 0, 0
706	001620	030722	.WORD	DF1	: 0, 0, 0, 0
707			:ITEM 26		
708			.WORD	EM26	:LMA CONTROL BITS INCORRECT
709	001622	024343	.WORD	DH24	:TESTNO ERR PC LMAEXP LMARCV
710	001624	027154	.WORD	DT26	:TESTNO,\$ERRPC,\$TMP0,\$REG2,0
711	001626	030474	.WORD	DF1	: 0, 0, 0, 0
712	001630	030722	.WORD	DF1	: 0, 0, 0, 0
713			:ITEM 27		
714			.WORD	EM27	:FORCE JUMPER BIT FAILS TO REVERT MAP REGISTER STATUS TO DEFAULT
715	001632	024376	.WORD	DH27	:TESTNO ERR PC LMARCV KIPAR4
716	001634	027213	.WORD	DT27	:TESTNO,\$ERRPC,\$TMP0,KIPAR4,0
717	001636	030506	.WORD	DF1	: 0, 0, 0
718	001640	030722	.WORD	DF1	: 0, 0, 0

719
720 001642 024476
721 001644 027252
722 001646 030520
723 001650 030722

:ITEM 30

.WORD EM30
.WORD DH30
.WORD DT30
.WORD DF1

:KIPAR5 NOT LOADED PROPERLY - SKIPPING NEXT TEST
:TESTNO ERR PC PR5EXP PR5RCV
:TESTNO,\$ERRPC,\$TMP5,KIPAR5,0
: 0, 0, 0, 0


```

777
778 001752 025377      ; ITEM 211
779                                .WORD  EM211  ;RELOCATION THROUGH THE MAP WAS NOT CORRECT, CARRY PROPAGATION
780 001754 030031      ;TEST BEING RUN OVER UNIBUS
781                                .WORD  DH211  ;CORRECT EXPECTD RECEIVD
782 001756 030660      ;ADDRESS DATA FROM UB TESTNO ERR PC
783 001760 030772      ;EADRES,$REG3,$REG2,TESTNO,$ERRPC,0
784                                .WORD  DF201  ; 2, 0, 0, 0, 0
785
786 001762 025535      ; ITEM 212
787                                .WORD  EM212  ;MAIN MEMORY TIME OUT OVER THE UNIBUS DID NOT OCCUR PROPERLY.
788 001764 030031      ;TEST BEING RUN OVER UNIBUS
789                                .WORD  DH211  ;CONDITN CONDITN
790 001766 030660      ;EXPECTD RECEIVD TESTNO ERR PC
791 001770 030772      ;CPUEXP,PCPUER,TESTNO,$ERRPC,0
792                                .WORD  DT211  ; 0, 0, 0, 0
793                                .WORD  DF201
794
795 001772 025670      ; ITEM 213
796 001774 030213      ;MAP REGISTER ENABLED WHEN DDW SAYS IT SHOULD BE DISABLED
797 001776 030706      ;TESTNO ERR PC REG NO DDWDAT DDWADR
798 002000 030722      ;TESTNO,$ERRPC,$TMP0,$TMP1,$REG5,0
799                                .WORD  DT213  ; 0, 0, 0, 0, 0
800                                .WORD  DF1
801
802 002002 025761      ; ITEM 214
803 002004 030213      ;MAP REGISTER DISABLED WHEN DDW SAYS IT SHOULD BE ENABLED
804 002006 030706      ;TESTNO ERR PC REG NO DDWDAT DDWADR
805 002010 030722      ;TESTNO,$ERRPC,$TMP0,$TMP1,$REG5,0
806                                .WORD  DT213  ; 0, 0, 0, 0, 0
807                                .WORD  DF1
  
```

804		.SBTTL	SOFTWARE SWITCH REGISTER LOCATION	
805	002012	.\$Y=.	:SAVE ADDRESS LOCATION	
806	000176	.=176	:ADDRESS TO SOFTWARE SWITCH REGISTER LOCATION	
807	000176 000000	\$\$SWR: .WORD 0	:LOCATION FOR SOFTWARE SWITCH REGISTER	
808	002012	.=.\$Y	:RETURN TO PREVIOUS ADDRESS LOCATION	

809
 810
 811
 812
 813
 814
 815
 816
 817
 818
 819
 820
 821
 822
 823
 824
 825
 826
 827
 828
 829
 830
 831
 832
 833
 834
 835
 836
 837
 838
 839
 840
 841
 842
 843
 844
 845
 846
 847
 848
 849
 850
 851 002012 010046
 852 002014 005000
 853 002016 113700 001112
 854 002022 001004
 855 002024 013746 001114
 856 002030 104402
 857 002032 000564
 858 002034 005300
 859 002036 072027 000003

```

.SBTTL ERROR MESSAGE TYPE OUT ROUTINE
*****
*
* THIS SUBROUTINE IS CALLED BY THE ERROR HANDLER TO TYPE
* THE ERROR MESSAGES. IT PICKS UP THE ITEM BYTE ($ITEMB) NUMBER
* AND USES THAT TO INDEX THROUGH THE ERROR TABLE. THE ERROR
* TABLE STARTS AT '$ERRTB' AND HAS FOUR (4) POINTERS FOR EACH
* ENTRY, 'EM', 'DH', 'DT', 'DF'. THE 'EM' POINTS TO THE ERROR
* MESSAGE WHICH IS AN ASCIZ STRING. THE 'DH' POINTS TO THE DATA
* HEADER WHICH IS ANOTHER ASCIZ STRING. THE 'DT' POINTS TO THE
* DATA TABLE WHICH IS A GROUP OF WORDS CONTAINING THE ADDRESSES
* OF THE DATA TO BE TYPED. THE FORMAT OF THIS DATA IS
* CONTROLLED BY THE 'DF' WHICH IS THE POINTER TO THE DATA FORMAT.
* THE DATA FORMAT IS A GROUP OF BYTES WHICH CONTAIN NUMBERS
* THAT CORRESPOND TO DIFFERENT TYPING FORMATS.
*
* 0 -16 BIT OCTAL FORMAT
* 1 -DECIMAL FORMAT
* 2 -22 BIT OCTAL FORMAT. DATA IS LOWER 16 BITS OF THE
* PHYSICAL ADDRESS, UPPER 6 BITS ARE ADJACENT TO LOWER 16
* 3 -22 BIT OCTAL FORMAT. DATA IS THE 16 BIT VIRTUAL
* ADDRESS IN KERNEL I-SPACE.
* 4 -18 BIT OCTAL FORMAT. DATA IS A 16 BIT NUMBER THAT
* WILL BE CONVERTED INTO A UNIBUS ADDRESS BY LEFT
* SHIFTING IT 6 BITS.
* 5 -16 BIT OCTAL, SUPPRESS LEADING ZEROS
* 6 -16 BIT DECIMAL, SUPPRESS SPACES
*
* IF YOU SHOULD HAVE A NEED TO JUST TYPE A STRING OF
* NUMBERS, SET UP YOUR CODE THIS WAY:
*
* MOV #CONTINUE,-(SP) ;MOVE THE ADDRESS OF THE INSTRUCTION AFTER THE
* ;JUMP TO THE STACK
*
* MOV R0,-(SP) ;SAVE R0
* MOV R1,-(SP) ;AND R1 ON THE STACK
* MOV DTNAME,R0 ;MOVE THE ADDRESS OF THE DATA TABLE TO R0
* JMP TYPDAT ;SUBROUTINE IDENTIFIED IN CENTER OF THIS ROUTINE
*
* CONTINUE: NEXT INSTRUCTION
* AT A CONVENIENT SPOT, ALLOCATE THE FOLLOWING:
*
* *DTNAME: .WORD DTLIST,DFNAME ;IDENTIFY THE LIST NAME AND DATA FORMAT BELOW
* *DFNAME: .BYTE N,N,N,N,ETC. ;CONSTRUCT YOUR OWN DATA FORMAT LINE
*
* .EVEN
* *DTLIST: VAR1,VAR2,VAR3,VAR4,.....,$CRLF,0 ;VARIABLES YOU WANT TYPED
*
*****
ERTYPE: MOV R0,-(KSP) ;SAVE R0 ON STACK
CLR R0 ;CLEAR R0
MOVB $ITEMB,R0 ;PUT ITEM NUMBER IN R0
BNE 1$ ;BRANCH IF IT IS NON-ZERO
MOV $ERRPC,-(KSP) ;PUT ERROR PC ON STACK FOR TYPING
TYPOC ;TYPE FAILING PC
BR 13$ ;GO TO RETURN
1$: DEC R0 ;ADJUST ITEM NUMBER TO BE A POINTER
ASH #3,R0 ;LEFT SHIFT ITEM NO. 3 PLACES
    
```

860 002042 100041
861 002044 023727 001302 000020
862 002052 002410
863 002054 001404
864 002056 062766 000004 000002
865 002064 000547

BPL 22\$
CMP ERRCNT,#20
BLT 40\$
BEQ 41\$
ADD #4,2(KSP)
BR 13\$

G 4

:BRANCH IF ITEM NUMBER IS LESS THAN 20
:;* SEE IF 20 (OCTAL) ERRORS HAVE PRINTED SEQ 0045
:;* BRANCH TO PRINT THE ERROR IF LESS
:;* BRANCH TO TYPE NO MORE DATA LINES IF EQUAL
:;* CORRECT PC RETURN TO RETURN AFTER <CRLF> PRINT
:;* GO TO RETURN

```

866 002066 104400 007260      41$:  TYPE      ,NOMORE      ;:* TYPE MESSAGE TO ANNOUNCE NO MORE PRINTING OF ERRORS
867 002072 000544              BR      13$          ;:* GO TO RETURN
868 002074 022737 000001 001302 40$:  CMP      #1,ERRCNT    ;:* SEE IF THIS IS THE FIRST ERROR
869 002102 001415              BEQ      21$          ;:* BRANCH IF IT WAS AND GO TYPE ERROR MESSAGE
870 002104 032777 000200 177036  BIT      #SW7,@SWR    ;SEE IF SWITCH 7 IS UP
871 002112 001404              BEQ      20$          ;BRANCH IF SWITCH NOT UP AND TYPE DATA
872 002114 062766 000004 000002  ADD      #4,2(KSP)   ;SKIP 'TYPE , $CRLF' IF SW 7 IS UP
873                                ;INHIBIT MULTIPLE ERROR TYPEOUTS
874 002122 000530              BR      13$          ;BRANCH TO EXIT
875 002124 042700 177400      20$:  BIC      #177400,RO   ;CLEAR UPPER BYTE OF RO
876 002130 062700 001656      ADD      #ER200+4,RO  ;POINT TO DATA TABLE ENTRY
877 002134 000426              BR      5$           ;GO TYPE DATA TABLE
878 002136 042700 177000      21$:  BIC      #177000,RO   ;CLEAR UPPER BYTE OF RO
879 002142 062700 000300      ADD      #<ER200-$ERRTB>,RO ;ADD DIFFERENCE BETWEEN
880                                ;ITEM 1 AND ITEM 201
881                                ;:GET POINTER TO ERROR MESSAGE AND TYPE IT
882                                ;:IF THE POINTER IS NOT ZERO
883 002146 104400 001221      22$:  TYPE      , $CRLF      ;TYPE A <CRLF>
884 002152 062700 001352      ADD      #$ERRTB,RO   ;ADD BASE OF ERROR TABLE
885 002156 012037 002166      MOV      (RO)+,2$    ;P M MESSAGE POINTER IN TYPE STATEMENT
886 002162 001404              BEQ      3$           ;BRANCH IF NO ERROR MESSAGE
887 002164 104400              TYPE      ;TYPE ERROR MESSAGE
888 002166 000000      2$:  .WORD    0           ;POINTER TO ERROR MESSAGE
889 002170 104400 001221      TYPE      , $CRLF    ;TYPE CRLF
890                                ;:GET THE POINTER TO THE DATA HEADER AND
891                                ;:TYPE IT IF THE POINTER IS NOT ZERO
892 002174 012037 002204      3$:  MOV      (RO)+,4$    ;PUT HEADER POINTER IN TYPE STATEMENT
893 002200 001404              BEQ      5$           ;BRANCH IF NO DATA HEADER
894 002202 104400              TYPE      ;TYPE THE DATA HEADER
895 002204 000000      4$:  .WORD    0           ;POINTER TO DATA HEADER
896 002206 104400 001221      TYPE      , $CRLF    ;TYPE CRLF
897                                ;:THIS IS THE START OF THE DATA OUTPUT IF THE
898                                ;:DATA POINTER IS NOT ZERO. RO POINTS TO THE
899                                ;:DATA FORMAT, R1 POINTS TO THE ADDRESS OF
900                                ;:THE DATA WORDS.
901 002212 010146      5$:  MOV      R1,-(KSP)   ;SAVE R1 ON THE STACK
902                                TYPDAT=.
903 002214 012001              MOV      (RO)+,R1    ;PUT DATA TABLE POINTER IN R1
904 002216 001471              BEQ      12$          ;BRANCH IF NO DATA TABLE
905 002220 012000              MOV      (RO)+,RO    ;PICK UP DATA FORMAT POINTER
906 002222 105710      6$:  TSTB     (RO)        ;IS THIS WORD OCTAL
907 002224 001003              BNE      7$           ;BRANCH IF NOT 16-BIT OCTAL
908                                ;:WORD IS 16 BIT OCTAL FORMAT (DF = 0)
909 002226 013146              MOV      @(R1)+,-(KSP) ;PUSH NEXT 16-BIT WORD ON STACK
910 002230 104402              TYPOC     ;TYPE THE WORD ON STACK AS 16 BIT OCTAL
911 002232 000456              BR      11$          ;GET READY FOR NEXT WORD
912 002234 122710 000001      7$:  CMPB     #1,(RO)     ;IS THE WORD DECIMAL
913 002240 001003              BNE      8$           ;BRANCH IF NOT DECIMAL
914                                ;:WORD IS DECIMAL FORMAT (DF = 1)
915 002242 013146              MOV      @(R1)+,-(KSP) ;PUSH NEXT 16-BIT WORD ON STACK
916 002244 104410              TYPDS     ;TYPE THE WORD ON STACK AS DECIMAL

```

917 002246 000450
918 002250 122710 000002
919 002254 001012
920
921 002256 012146
922 002260 004737 004166

8\$:

BR 11\$
CMPB #2,(R0)
BNE 9\$
MOV (R1)+,-(KSP)
JSR PC,\$DB20

I 4

;GET READY FOR NEXT WORD
;IS WORD 22-BIT PHYSICAL ADDRESS
;BRANCH IF NOT 22-BIT PHYSICAL ADDR
;:WORD IS 22-BIT PHYSICAL FORMAT (DF = 2)
;PUSH NEXT 16-BIT WORD ON STACK
;CONVERT NUMBER TO OCTAL ASCIZ

SEQ 0047

CR
CC


```

923 002264 062716 000003      ADD    #5,(KSP)      ;ONLY WANT 8 DIGITS
924 002270 012637 002276      MOV    (KSP)+,30$   ;PUT POINTER AFTER 'TYPE' CALL
925 002274 104400              TYPE                   ;TYPE ASCIZ STRING
926 002276 000000              30$: .WORD    0      ;WORD HOLDS POINTER TO ASCIZ STRING
927 002300 000433              BR     11$           ;GET READY FOR NEXT WORD
928 002302 122710 000003      9$:  CMPB   #3,(RO)   ;IS THIS A 16-BIT VIRTUAL ADDRESS
929 002306 001004              BNE    10$           ;BRANCH IF NOT 16-BIT VIRT. ADDR.
930                                ;:WORD IS 22-BIT VIRTUAL ADDRESS FORMAT
931                                ;:KERNEL I-SPACE ASSUMED. (DF = 3)
932 002310 013146              MOV    @(R1)+,-(KSP) ;PUSH NEXT 16-BIT WORD ON STACK
933 002312 004737 002414      JSR    PC,TYPVAD     ;GO TYPE 22-BIT ADDRESS FROM 16-BIT V.A.
934 002316 000424              BR     11$           ;GET READY FOR NEXT WORD
935 002320 122710 000004      10$: CMPB   #4,(RO)   ;IS THIS A 16 BIT NUMBER TO BE CONVERTED TO
936                                ;AN 18 BIT UNIBUS ADDRESS LEFT SHIFTED 6?
937 002324 001003              BNE    100$          ;SKIP OVER FORMAT 4 ROUTINE IF NOT
938                                ;:WORD IS FORMAT 4. DATA WORD IS A UNIBUS
939                                ;:ADDRESS OUTPUT WILL BE 18-BITS WORD LEFT SHIFTED 6.
940
941 002326 004737 002522      JSR    PC,UBADDR     ;CONVERT TO 18-BIT UNIBUS ADDR AND TYPE
942 002332 000416              BR     11$           ;GET READY FOR NEXT WORD
943 002334 122710 000005      100$: CMPB   #5,(RO)   ;IS THIS A 16 BIT NUMBER TO BE PRINTED AS
944                                ;OCTAL WITH LEADING ZEROS SUPPRESSED?
945 002340 001004              BNE    110$          ;BRANCH TO DECIMAL LEADING SPACES SUPPRESS ROUTINE
946                                ;:WORD IS FORMAT 5. DATA WORD IS TO BE
947                                ;:PRINTED IN OCTAL, LEADING ZEROS SUPPRESSED.
948 002342 013146              MOV    @(R1)+,-(KSP) ;PUSH NEXT 16-BIT WORD ON STACK
949 002344 104404              TYPOS                   ;GO TYPE OCTAL SUPPRESS LEADING ZEROS
950 002346 006                .BYTE 6                ;TYPE 6 DIGITS AND
951 002347 000                .BYTE 0                ;SUPPRESS LEADING ZEROS
952 002350 000407              BR     11$           ;GET READY FOR NEXT WORD
953 002352              110$:                ;:WORD IS FORMAT 6. DATA WORD IS TO BE
954                                ;:PRINTED IN DECIMAL, LEADING SPACES SUPPRESSED.
955 002352 013146              MOV    @(R1)+,-(KSP) ;PUSH NEXT 16-BIT WORD ON STACK
956 002354 112737 000001 002413  MOVB   #1,SPSUPP     ;SET FLAG TO SUPPRESS LEADING SPACES
957 002362 104410              TYPDS                   ;CLEAR THE LEADING SPACES SUPPRESS FLAG
958 002364 105037 002413      CLRB   SPSUPP        ;POINT TO NEXT FORMAT BYTE
959 002370 005200              11$: INC     RO        ;TYPE TWO SPACES
960 002372 104400 002410      TYPE   ,32$          ;IS THERE ANOTHER WORD?
961 002376 005711              TST    (R1)           ;BRANCH IF NOT ALL DONE
962 002400 001310              BNE    6$             ;RESTORE R1
963 002402 012601              12$: MOV    (KSP)+,R1  ;RESTORE RO
964 002404 012600              13$: MOV    (KSP)+,RO  ;RETURN TO ERROR ROUTINE
965 002406 000207              RTS    PC             ;TWO SPACES
966 002410 040 040 000 32$: .ASCIZ ? ? ;LEADING ZEROS SUPPRESS FLAG LOCATION
967 002413 000  SPSUPP: .BYTE 0

```

```

968
969
970
971
972
973
974
975
976
977
978
979
980 002414 104412
981 002416 016601 000002
982 002422 005000
983 002424 073027 000003
984 002430 006300
985 002432 006001
986 002434 006001
987 002436 006001
988 002440 062700 172340
989 002444 011003
990 002446 005002
991 002450 073227 000006
992 002454 060103
993 002456 005502
994 002460 010237 001226
995 002464 010337 001224
996 002470 012746 001224
997 002474 004737 004166
998 002500 062716 000003
999 002504 012637 002512
1000 002510 104400
1001 002512 000000
1002
1003 002514 104414
1004 002516 012616
1005 002520 000207
    
```

```

.SBTTL CONVERT 16-BIT VIRTUAL ADDRESS TO 22-BIT PHYSICAL ADDRESS
:*****
:
: THIS ROUTINE IS CALLED BY A 'JSR PC' AFTER THE VIRTUAL ADDRESS
: IS PUSHED ON THE KERNEL STACK. THE V.A. IS THEN LOADED INTO
: R1 AND THE UPPER 3 BITS ARE SHIFTED INTO R0 TO SELECT THE
: CORRECT KERNEL I-SPACE PAR. THE LOWER 12 BITS OF THE VIRTUAL
: ADDRESS ARE ADDED TO THE PAR AS THEY ARE BY MEMORY MANAGEMENT
: AND THE PHYSICAL ADDRESS IS SAVED IN MEMORY TO BE CONVERTED
: TO ASCIZ AND TYPED.
:*****
TYPVAD: SAVREG ;SAVE ALL REGISTERS
MOV 2(KSP),R1 ;PUT VIRTUAL ADDR IN R1
CLR R0 ;CLEAR R0 FOR CALCULATIONS
ASHC #3,R0 ;LEFT SHIFT R0,R1 3 PLACES
ASL R0 ;LEFT SHIFT R0 ONE MORE PLACE
ROR R1 ;RIGHT SHIFT R1 SO OFFSET IS CORRECT
ROR R1 ;RIGHT SHIFT R1
ROR R1 ;RIGHT SHIFT R1
ADD #KIPAR0,R0 ;FORM DESIRED PAR ADDR IN R0
MOV (R0),R3 ;PUT CONTENTS OF PAR IN R3
CLR R2 ;CLEAR R2 FOR PHYSICAL ADDR CALCULATIONS
ASHC #6,R2 ;LEFT SHIFT <R2,R3> 6 PLACES
ADD R1,R3 ;ADD OFFSET IN R1 TO BASE IN R3
ADC R2 ;ADD ANY POSSIBLE CARRY TO UPPER 6 BITS
MOV R2,PADRSR ;PUT UPPER 6 BITS OF ADDR IN CORE
MOV R3,PADRSL ;PUT LOWER 16 BITS OF ADDR IN CORE
MOV #PADRSL,-(KSP) ;PUT POINTER TO LOWER 16 BITS ON STACK
JSR PC,$DB20 ;CONVERT NUMBER TO OCTAL ASCIZ
ADD #3,(KSP) ;ONLY TYPE 8 DIGITS
MOV (KSP)+,3$ ;PUT POINTER AFTER TYPE INST
TYPE ;TYPE THE 22-BIT VIRTUAL ADDRESS
3$: .WORD 0 ;THIS WORD HOLDS THE POINTER TO
;THE ASCIZ STRING
RESREG ;RESTORE ALL THE REGISTERS
MOV (KSP)+,(KSP) ;LEAVE ONLY RETURN ADDR ON STACK
RTS PC ;RETURN TO ERROR HANDLER
    
```

```

1006      .SBTTL  SUBROUTINE TO CONVERT WORD TO A UNIBUS ADDRESS AND TYPE
1007      :*****
1008      :*THIS SUBROUTINE IS USED TO CONVERT THE A WORD PUSHED
1009      :*ON THE STACK INTO A UNIBUS ADDRESS AND TYPE IT AS A
1010      :*6 DIGIT NUMBER.  IT USES R1 & R0 AND LEAVES
1011      :*ALL OTHER REGISTERS UNCHANGED.
1012      :*****
1013 002522 104412  UBADDR: SAVREG
1014 002524 016601 000002      MOV      2(KSP),R1      ;LOAD 16 BIT ADDRESS INTO R1
1015 002530 005000      CLR      R0           ;CLEAR R0 FOR CALCULATIONS
1016 002532 073027 000006      ASHC   #6,R0         ;LEFT SHIFT <R0:R1> 6 PLACES
1017 002536 010137 001224      MOV     R1,PADRSL     ;PUT LOWER 16 BITS IN PADRSL
1018 002542 010037 001226      MOV     R0,PADRSH     ;PUT UPPER 6 BITS IN PADRSH
1019 002546 012746 001224      MOV     #PADRSL,-(KSP) ;PUSH POINTER TO WORDS ON STACK
1020 002552 004737 004166      JSR     PC,$DB20      ;JUMP TO CONVERT ROUTINE
1021 002556 062716 000005      ADD     #5,(KSP)      ;ONLY USE LOWER 6 CHARS.
1022 002562 012637 002570      MOV     (KSP)+,3$     ;PUT POINTER AFTER TYPE CALL.
1023 002566 104400      TYPE
1024 002570 000000 3$:      .WORD  0             ;HOLDS POINTER TO FIRST CHAR.
1025 002572 104414      RESREG
1026 002574 012616      MOV     (KSP)+,(KSP)  ;LEAVE ONLY RETURN ADDRESS ON STACK
1027 002576 000207      RTS      PC          ;RETURN TO ERROR TYPE ROUTINE.
  
```

1028

.SBTTL SAVE AND RESTORE R0-R5 ROUTINES
 ;*****

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

002600
 002600 010046
 002602 010146
 002604 010246
 002606 010346
 002610 010446
 002612 010546
 002614 016646 000022
 002620 016646 000022
 002624 016646 000022
 002630 016646 000022
 002634 000002

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

*RESTORE R0-R5

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

002636
 002636 012666 000022
 002642 012666 000022
 002646 012666 000022
 002652 012666 000022
 002656 012605
 002660 012604
 002662 012603
 002664 012602
 002666 012601
 002670 012600
 002672 000002

1029

```

.SBTTL TYPE ROUTINE
*****
*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
*NOTE1: $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
*NOTE2: $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
*NOTE3: $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
*
*CALL:
*1) USING A TRAP INSTRUCTION
*   TYPE      ,MESADR      ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
*OR
*   TYPE
*   MESADR
*
*2) USING A JSR INSTRUCTION
*   MOV      PS,-(SP)      ;;PUSH PROCESSOR STATUS WORD ON THE STACK
*   JSR      PC,$TYPE     ;;CALL TYPE ROUTINE
*   MESADDR  ;;FIRST ADDRESS OF MESSAGE
$TYPE:  TSTB  $TPFLG      ;;IS THERE A TERMINAL?
        BPL   1$         ;;BR IF YES
        HALT  ;;HALT HERE IF NO TERMINAL
        BR    3$         ;;LEAVE
1$:     MOV   RO,-(SP)    ;;SAVE RO
        MOV   @2(SP),RO  ;;GET ADDRESS OF ASCIZ STRING
        BITB #40,$ENVM  ;;&& SUPRESS TYPEOUTS?
        BNE  60$        ;;&& YES
2$:     MOVB (RO)+,-(SP) ;;PUSH CHARACTER TO BE TYPED ONTO STACK
        BNE  4$         ;;BR IF IT ISN'T THE TERMINATOR
        TST  (SP)+      ;;IF TERMINATOR POP IT OFF THE STACK
60$:    MOV   (SP)+,RO   ;;&& RESTORE RO
3$:     ADD   #2,(SP)   ;;ADJUST RETURN PC
        RTI  ;;RETURN
4$:     CMPB #HT,(SP)  ;;BRANCH IF <HT>
        BEQ  8$
        CMPB #CRLF,(SP) ;;BRANCH IF NOT
        BNE  5$
        TST  (SP)+      ;;POP <CR><LF> EQUIV
        TYPE ,SCRLF
        BR   2$         ;;GET NEXT CHARACTER
5$:     JSR  PC,$TYPEC  ;;GO TYPE THIS CHARACTER
6$:     CMPB $FILLC,(SP)+ ;;IS IT TIME FOR FILLER CHARS.?
        BNE  2$         ;;IF NO GO GET NEXT CHAR.
        MOV  $NULL,-(SP) ;;GET # OF FILLER CHARS. NEEDED
        ;;AND THE NULL CHAR.
7$:     DECB 1(SP)     ;;DOES A NULL NEED TO BE TYPED?
        BLT  6$         ;;BR IF NO--GO POP THE NULL OFF OF STACK
        JSR  PC,$TYPEC  ;;GO TYPE A NULL
        DECB $CHARCNT  ;;DON'T COUNT THE NULL AS A CHARACTER
        BR   7$         ;;LOOP
;;HORIZONTAL TAB PROCESSOR

```

```

002674 105737 001147
002700 100002
002702 000000
002704 000413
002706 010046
002710 017600 000002
002714 132737 000040 020035
002722 001003
002724 112046
002726 001005
002730 005726
002732 012600
002734 062716 000002
002740 000002
002742 122716 000011
002746 001426
002750 122716 000200
002754 001004
002756 005726
002760 104400 001221
002764 000757
002766 004737 003050
002772 123726 001146
002776 001352
003000 013746 001144
003004 105366 000001
003010 002770
003012 004737 003050
003016 105337 003200
003022 000770

```

003024 112716 000040
003030 004737 003050
003034 132737 000007 003200
003042 001372
003044 005726
003046 000726

8\$:
9\$:

MOVB #' (SP)
JSR PC,\$TYPEC
BITB #7,\$CHARCNT
BNE 9\$
TST (SP)+
BR 2\$

B 5

::REPLACE TAB WITH SPACE
::TYPE A SPACE
::BRANCH IF NOT AT
::TAB STOP
::POP SPACE OFF STACK
::GET NEXT CHARACTER

SEQ 0053

```

003050 105777 176064          $TYPEC: TSTB  @STPS          ;;WAIT UNTIL PRINTER IS READY
003054 100375                BPL          $TYPEC
003056 116677 000002 176056  MOVB  2(SP),@STPB  ;;LOAD CHAR TO BE TYPED INTO DATA REG.
003064 105777 176044          TSTB  @STKS          ;;SEE IF KEYBOARD IS TALKING
003070 100027                BPL          2$      ;;BRANCH AROUND XON/XOFF ROUTINE IF NOT
003072 117737 176040 006140  MOVB  @STKB,CHARCT ;;PUSH CHARACTER ONTO STACK
003100 042737 177600 006140  BIC   #177600,CHARCT ;;BIT CLEAR TOP BYTE AND PARITY BIT
003106 022737 000023 006140  CMP   #23,CHARCT   ;;SEE IF IT IS A ^S
003114 001015                BNE          2$      ;;BR IF IT ISN'T
003116 105777 176012          3$:  TSTB  @STKS          ;;SEE IF KEYBOARD IS TALKING
003122 100375                BPL          3$      ;;BR BACK IF NOT READY
003124 017737 176006 006140  MOV   @STKB,CHARCT ;;PUSH NEXT CHARACTER ON STACK
003132 042737 177600 006140  BIC   #177600,CHARCT ;;BIT CLEAR TOP BYTE AND PARITY BIT
003140 022737 000021 006140  CMP   #21,CHARCT   ;;IS IT A ^Q
003146 001363                BNE          3$      ;;BR IF NOT
003150 122766 000015 000002  2$:  CMPB  #CR,2(SP)   ;;BRANCH IF
003156 001003                BNE          1$      ;;NOT <CR>
003160 105037 003200          CLRB  $CHARCNT     ;;
003164 000406                BR          $TYPEX  ;;EXIT
003166 122766 000012 000002  1$:  CMPB  #LF,2(SP)   ;;BRANCH IF
003174 001402                BEQ          $TYPEX  ;;<LF>
003176 105227                INCB  (PC)+        ;;INC SPACE
003200 000000          $CHARCNT: .WORD  0  ;;COUNT
003202 000207          $TYPEX: RTS      PC

```

1030

```

.SBTTL BINARY TO OCTAL (ASCII) AND TYPE
*****
*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
*OCTAL (ASCII) NUMBER AND TYPE IT.
*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
*CALL:
*   MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
*   TYPOS    ;;CALL FOR TYPEOUT
*   .BYTE   N              ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
*   .BYTE   M              ;;M=1 OR 0
*                               ;;1=TYPE LEADING ZEROS
*                               ;;0=SUPPRESS LEADING ZEROS
*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
*$TYPOS OR $TYPOC
*CALL:
*   MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
*   TYPON    ;;CALL FOR TYPEOUT
*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
*CALL:
*   MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
*   TYPOC    ;;CALL FOR TYPEOUT
003204 017646 000000 003427 $TYPOS: MOV @ (SP),-(SP) ;;PICKUP THE MODE
003210 116637 000001 003427 MOV 1(SP), $OFILL ;;LOAD ZERO FILL SWITCH
003216 112637 003431 MOV  (SP)+, $OMODE+1 ;;NUMBER OF DIGITS TO TYPE
003222 062716 000002 ADD #2, (SP) ;;ADJUST RETURN ADDRESS
003226 000406 BR $TYPON
003230 112737 000001 003427 $TYPOC: MOV #1, $OFILL ;;SET THE ZERO FILL SWITCH
003236 112737 000006 003431 MOV #6, $OMODE+1 ;;SET FOR SIX(6) DIGITS
003244 112737 000005 003426 $TYPON: MOV #5, $OCNT ;;SET THE ITERATION COUNT
003252 010346 MOV R3, -(SP) ;;SAVE R3
003254 010446 MOV R4, -(SP) ;;SAVE R4
003256 010546 MOV R5, -(SP) ;;SAVE R5
003260 113704 003431 MOV #5, R4 ;;GET THE NUMBER OF DIGITS TO TYPE
003264 005404 NEG R4
003266 062704 000006 ADD #6, R4 ;;SUBTRACT IT FOR MAX. ALLOWED
003272 110437 003430 MOV R4, $OMODE ;;SAVE IT FOR USE
003276 113704 003427 MOV $OFILL, R4 ;;GET THE ZERO FILL SWITCH
003302 016605 000012 MOV 12(SP), R5 ;;PICKUP THE INPUT NUMBER
003306 005003 CLR R3 ;;CLEAR THE OUTPUT WORD
003310 006105 1$: ROL R5 ;;ROTATE MSB INTO "C"
003312 000404 BR 3$ ;;GO DO MSB
003314 006105 2$: ROL R5 ;;FORM THIS DIGIT
003316 006105 ROL R5
003320 006105 ROL R5
003322 010503 MOV R5, R3
003324 006103 3$: ROL R3 ;;GET LSB OF THIS DIGIT
003326 105337 003430 DECB $OMODE ;;TYPE THIS DIGIT?
003332 100016 BPL 7$ ;;BR IF NO
003334 042703 177770 BIC #177770, R3 ;;GET RID OF JUNK
    
```


003340 001002
003342 005704
003344 001403
003346 005204
003350 052703 000060
003354 052703 000040

4\$: BNE 4\$
TST R4
BEQ 5\$
INC R4
5\$: BIS #'0,R3
BIS #' ,R3

E 5

:::TEST FOR 0
:::SUPPRESS THIS 0?
:::BR IF YES
:::DON'T SUPPRESS ANYMORE 0'S
:::MAKE THIS DIGIT ASCII
:::MAKE ASCII IF NOT ALREADY

SEQ 0056

003360	110337	003424		MOVB	R3,8\$::SAVE FOR TYPING
003364	104400	003424		TYPE	8\$::GO TYPE THIS DIGIT
003370	105337	003426	7\$:	DECB	\$OCNT	::COUNT BY 1
003374	003347			BGT	2\$::BR IF MORE TO DO
003376	002402			BLT	6\$::BR IF DONE
003400	005204			INC	R4	::INSURE LAST DIGIT ISN'T A BLANK
003402	000744			BR	2\$::GO DO THE LAST DIGIT
003404	012605		6\$:	MOV	(SP)+,R5	::RESTORE R5
003406	012604			MOV	(SP)+,R4	::RESTORE R4
003410	012603			MOV	(SP)+,R3	::RESTORE R3
003412	016666	000002 000004		MOV	2(SP),4(SP)	::SET THE STACK FOR RETURNING
003420	012616			MOV	(SP)+,(SP)	
003422	000002			RTI		::RETURN
003424	000		8\$:	.BYTE	0	::STORAGE FOR ASCII DIGIT
003425	000			.BYTE	0	::TERMINATOR FOR TYPE ROUTINE
003426	000		\$OCNT:	.BYTE	0	::OCTAL DIGIT COUNTER
003427	000		\$OFILL:	.BYTE	0	::ZERO FILL SWITCH
003430	000000		\$OMODE:	.WORD	0	::NUMBER OF DIGITS TO TYPE

1031

.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

003432				\$TYPDS:	MOV	R0,-(SP)	;;PUSH R0 ON STACK
003432	010046				MOV	R1,-(SP)	;;PUSH R1 ON STACK
003434	010146				MOV	R2,-(SP)	;;PUSH R2 ON STACK
003436	010246				MOV	R3,-(SP)	;;PUSH R3 ON STACK
003440	010346				MOV	R5,-(SP)	;;PUSH R5 ON STACK
003442	010546				MOV	#20200,-(SP)	;;SET BLANK SWITCH AND SIGN
003444	012746	020200			MOV	20(SP),R5	;;GET THE INPUT NUMBER
003450	016605	000020			BPL	1\$;;BR IF INPUT IS POS.
003454	100004				NEG	R5	;;MAKE THE BINARY NUMBER POS.
003456	005405				MOV	#'-,1(SP)	;;MAKE THE ASCII NUMBER NEG.
003460	112766	000055	000001	1\$:	CLR	R0	;;ZERO THE CONSTANTS INDEX
003466	005000				MOV	#\$DBLK,R3	;;SETUP THE OUTPUT POINTER
003470	012703	003714			MOV	#',(R3)+	;;SET THE FIRST CHARACTER TO A BLANK
003474	112723	000040		2\$:	CLR	R2	;;CLEAR THE BCD NUMBER
003500	005002				MOV	\$DTBL(R0),R1	;;GET THE CONSTANT
003502	016001	003704		3\$:	SUB	R1,R5	;;FORM THIS BCD DIGIT
003506	160105				BLT	4\$;;BR IF DONE
003510	002402				INC	R2	;;INCREASE THE BCD DIGIT BY 1
003512	005202				BR	3\$	
003514	000774				ADD	R1,R5	;;ADD BACK THE CONSTANT
003516	060105			4\$:	TST	R2	;;CHECK IF BCD DIGIT=0
003520	005702				BNE	5\$;;FALL THROUGH IF 0
003522	001002				TSTB	(SP)	;;STILL DOING LEADING 0'S?
003524	105716				BMI	7\$;;BR IF YES
003526	100407				ASLB	(SP)	;;MSD?
003530	106316			5\$:	BCC	6\$;;BR IF NO
003532	103003				MOV	1(SP),-1(R3)	;;YES--SET THE SIGN
003534	116663	000001	177777	6\$:	BIS	#'0,R2	;;MAKE THE BCD DIGIT ASCII
003542	052702	000060		7\$:	BIS	#',R2	;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
003546	052702	000040			MOV	R2,(R3)+	;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
003552	110223				TST	(R0)+	;;JUST INCREMENTING
003554	005720				CMP	R0,#10	;;CHECK THE TABLE INDEX
003556	020027	000010			BLT	2\$;;GO DO THE NEXT DIGIT
003562	002746				BGT	8\$;;GO TO EXIT
003564	003002				MOV	R5,R2	;;GET THE LSD
003566	010502				BR	6\$;;GO CHANGE TO ASCII
003570	000764			8\$:	TSTB	(SP)+	;;WAS THE LSD THE FIRST NON-ZERO?
003572	105726				BPL	9\$;;BR IF NO
003574	100003				MOV	-1(SP),-2(R3)	;;YES--SET THE SIGN FOR TYPING
003576	116663	177777	177776	9\$:	CLRB	(R3)	;;SET THE TERMINATOR
003604	105013						

003606 012605
003610 012603
003612 012602
003614 012601
003616 012600
003620 105737 002413

MOV (SP)+,R5
MOV (SP)+,R3
MOV (SP)+,R2
MOV (SP)+,R1
MOV (SP)+,R0
TSTB SPSUPP

H 5

::POP STACK INTO R5
::POP STACK INTO R3
::POP STACK INTO R2
::POP STACK INTO R1
::POP STACK INTO R0
::SEE IF LEADING SPACES ARE TO BE SUPPRESSED

SEQ 0059

```

003624 001420          BEQ      12$          ;: BRANCH TO TYPE THE NUMBER IF NOT
003626 010046          MOV      R0,-(SP)       ;: SAVE R0
003630 010146          MOV      R1,-(SP)       ;: SAVE R1
003632 012700 003714   MOV      #$DBLK,R0      ;: LOAD STARTING ADDRESS OF BYTES IN R0
003636 012701 003714   MOV      #$DBLK,R1      ;: LOAD STARTING ADDRESS OF BYTES IN R1
003642 122720 000040   10$:  CMPB    #'',(R0)+    ;: SEE IF A SPACE CHARACTER IS HERE
003646 001775          BEQ      10$          ;: BRANCH BACK FOR ANOTHER TRY IF SO
003650 005300          DEC      R0          ;: POINT R0 BACK TO FIRST NON-SPACE ASCII
003652 112021 11$:  MOVB    (R0)+,(R1)+    ;: MOVE THE NON-ZERO CHARACTER UP FRONT
003654 105710          TSTB    (R0)          ;: SEE IF TERMINATOR IS PRESENT
003656 001375          BNE     11$          ;: BRANCH TO MOVE ANOTHER CHARACTER IF NOT
003660 105011          CLRB   (R1)          ;: PLACE A TERMINATOR BYTE AFTER CHARACTERS
003662 012601          MOV     (SP)+,R1      ;: RESTORE R1
003664 012600          MOV     (SP)+,R0      ;: RESTORE R0
003666 104400 003714   12$:  TYPE    $DBLK      ;: NOW TYPE THE NUMBER
003672 016666 000002 000004 MOV     2(SP),4(SP)    ;: ADJUST THE STACK
003700 012616          MOV     (SP)+,(SP)
003702 000002          RTI
003704 023420          $DTBL: 10000.
003706 001750          1000.
003710 000144          100.
003712 000012          10.
003714          $DBLK: .BLKW 4
  
```

1032

.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.

003724 010046
003726 016600 000002
003732 005740
003734 111000
003736 016000 003744
003742 000200

\$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:

003744		\$TYPE	::CALL=TYPE	TRAP+0(104400)	TTY TYPEOUT ROUTINE
003744	002674	\$TYPOC	::CALL=TYPOC	TRAP+2(104402)	TYPE OCTAL NUMBER (WITH LEADING ZEROS)
003746	003230	\$TYPOS	::CALL=TYPOS	TRAP+4(104404)	TYPE OCTAL NUMBER (NO LEADING ZEROS)
003750	003204	\$TYPON	::CALL=TYPON	TRAP+6(104406)	TYPE OCTAL NUMBER (AS PER LAST CALL)
003752	003244	\$TYPDS	::CALL=TYPDS	TRAP+10(104410)	TYPE DECIMAL NUMBER (WITH SIGN)
003754	003432	\$SAVREG	::CALL=SAVREG	TRAP+12(104412)	SAVE R0-R5 ROUTINE
003756	002600	\$RESREG	::CALL=RESREG	TRAP+14(104414)	RESTORE R0-R5 ROUTINE
1033	003762	TBITOFF	::CALL=TBITO	TRAP+16(104416)	THIS WILL TURN OFF T BIT TRAPPING
1034	003764	TBITRESTORE	::CALL=TBITR	TRAP+20(104420)	THIS WILL RETURN THE T BIT TO PREVIOUS
1035	004306				
	004334				

1036

.SBTTL POWER DOWN AND UP ROUTINES

:POWER DOWN ROUTINE

```

003766 012737 004114 000024 $PWRDN: MOV    #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
003774 012737 000340 000026      MOV    #340,@#PWRVEC+2 ;;PRIO:7
004002 010046      MOV    R0,-(SP) ;;PUSH R0 ON STACK
004004 010146      MOV    R1,-(SP) ;;PUSH R1 ON STACK
004006 010246      MOV    R2,-(SP) ;;PUSH R2 ON STACK
004010 010346      MOV    R3,-(SP) ;;PUSH R3 ON STACK
004012 010446      MOV    R4,-(SP) ;;PUSH R4 ON STACK
004014 010546      MOV    R5,-(SP) ;;PUSH R5 ON STACK
004016 010637 004120      MOV    SP,$SAVR6 ;;SAVE SP
004022 012737 004034 000024      MOV    #$PWRUP,@#PWRVEC ;;SET UP VECTOR
004030 000000      HALT
004032 000776      BR     -2 ;;HANG UP
    
```

:POWER UP ROUTINE

```

004034 013706 004120 $PWRUP: MOV    $SAVR6,SP ;;GET SP
004040 005037 004120      CLR    $SAVR6 ;;WAIT LOOP FOR THE TTY
004044 005237 004120 1$: INC    $SAVR6 ;;WAIT FOR THE INC
004050 001375      BNE    1$ ;;OF WORD
004052 012605      MOV    (SP)+,R5 ;;POP STACK INTO R5
004054 012604      MOV    (SP)+,R4 ;;POP STACK INTO R4
004056 012603      MOV    (SP)+,R3 ;;POP STACK INTO R3
004060 012602      MOV    (SP)+,R2 ;;POP STACK INTO R2
004062 012601      MOV    (SP)+,R1 ;;POP STACK INTO R1
004064 012600      MOV    (SP)+,R0 ;;POP STACK INTO R0
004066 012737 003766 000024      MOV    #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
004074 012737 000340 000026      MOV    #340,@#PWRVEC+2 ;;PRIO:7
004102 104400      TYPE   PWRMSG ;;REPORT THE POWER FAILURE
004104 004122 $PWRMG: .WORD PWRMSG ;;POWER FAIL MESSAGE POINTER
004106 012716      MOV    (PC)+,(SP) ;;RESTART AT START
004110 010000 $PWRAD: .WORD START ;;RESTART ADDRESS
004112 000002      RTI
004114 000000 $ILLUP: HALT ;;THE POWER UP SEQUENCE WAS STARTED
004116 000776      BR     -2 ;; BEFORE THE POWER DOWN WAS COMPLETE
004120 000000 $SAVR6: 0 ;;PUT THE SP HERE
1037 004122 012 015 120 PWRMSG: .ASCIZ <12><15>?POWER FAILURE, RESTARTING PROGRAM?
1038      .EVEN
    
```


1039

```

.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
$DB20: SAVREG      ;; SAVE ALL REGISTERS
MOV      2(SP),R1      ;; PICKUP THE POINTER TO LOW WORD
MOV      #SOCTVL+13.,R5 ;; POINTER TO DATA TABLE
MOV      #12.,R4      ;; DO ELEVEN CHARACTERS
MOV      #^C7,R3      ;; MASK
MOV      (R1)+,R0     ;; LOWER WORD
MOV      (R1)+,R1     ;; HIGH WORD
CLR      R2           ;; TERMINATOR
1$:      MOVVB   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
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
    
```

```

004166 104412
004170 016601 000002
004174 012705 004305
004200 012704 000014
004204 012703 177770
004210 012100
004212 012101
004214 005002
004216 110245
004220 010002
004222 005304
004224 003007
004226 001405
004230 005205
004232 010566 000002
004236 104414
004240 000207
004242 006203
004244 006001
004246 006000
004250 006001
004252 006000
004254 006001
004256 006000
004260 040302
004262 062702 000060
004266 000753
004270
    
```

```

1041 .SBTTL TURN OFF AND SAVE T-BIT
1042 :*****
1043 :*****
1044 :*                **SUBROUTINES UNIQUE TO THIS PROGRAM**
1045 :*****
1046 :*****
1047 :
1048 :*****
1049 :*
1050 :* THIS SUBROUTINE IS REACHED BY THE TRAP CALL 'TBITO', IT IS
1051 :* USED TO TURN OFF THE T-BIT IF IT IS ON. THE PROCESSOR STATUS
1052 :* IS SAVED IN 'OLDPSW' SO THAT THE T-BIT CAN BE RESTORED TO ITS
1053 :* PREVIOUS STATUS WHEN CONDITIONS WARRANT.
1054 :*
1055 :*****
1056 000020          TBIT=BIT4          ;T-BIT IS BIT04 IN PROC. STATUS
1057 004306 032766 000020 000002 TBITOF: BIT    #TBIT,2(KSP) ;IS THE T-BIT ON?
1058 004314 001406          BEQ      1$          ;BRANCH TO EXIT IF IT IS NOT ON
1059 004316 016637 000002 001332          MOV     2(KSP),OLDPSW ;SAVE OLD PSW FOR RESTORING T BIT
1060 004324 042766 000020 000002          BIC    #TBIT,2(KSP) ;CLEAR T BIT
1061 004332 000006          1$:      RTT          ;RETURN TO PROGRAM
    
```

1062
1063
1064
1065
1066
1067
1068
1069
1070
1071

.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'.
: *
:*****

1072 004334 013766 001332 000002
1073 004342 042737 000020 001332
1074
1075 004350 000006

TBITRE: 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

```
1076 .SBTTL SUBROUTINE TO CLEAR ALL OF THE MAP REGISTERS
1077 :*****
1078 :*
1079 :* THIS SUBROUTINE CLEARS ALL OF THE MAP REGISTERS IF MAPPING IS
1080 :* DISABLED BY LOADING THE ADDRESS OF MAPLO0 INTO R3 AND THEN
1081 :* CLEARING THE REGISTER POINTED TO BY R3 UNTIL R3 POINTS ABOVE
1082 :* MAPH37. IF MAPPING IS ENABLED, ALL REGISTERS EXCEPT MAPL1
1083 :* IS CLEARED. THE LOWER WORD OF MAPL1 RECEIVES 20000. THIS IS
1084 :* SO APT CAN PROPERLY MONITOR THE PROGRESS OF THE DIAGNOSTIC.
1085 :*
1086 :*****
1087 004352 012703 170200 CLRMAP: MOV #MAPLO,R3 ;PUT FIRST MAP ADDR IN R3
1088 004356 005023 CLR (R3)+ ;CLEAR MAPLO
1089 004360 005023 CLR (R3)+ ;CLEAR MAPLO+2
1090 004362 032737 000040 172516 BIT #BIT5,MMR3 ;SEE IF MAPPING IS ENABLED
1091 004370 001402 BEQ 1$ ;BRANCH TO CLEAR ALL IF NOT ENABLED
1092 004372 012723 020000 MOV #20000,(R3)+ ;LOAD 20000 INTO MAPL1 FOR POSSIBLE APT USE
1093 004376 005023 1$: CLR (R3)+ ;CLEAR MAP REGISTERS
1094 004400 022703 170400 CMP #MAPH37+2,R3 ;SEE IF LAST ADDR+2 IS IN R3
1095 004404 001374 BNE 1$ ;BRANCH IF NOT DONE YET
1096 004406 000207 RTS PC ;RETURN TO MAIN PROGRAM
```

```

1097 .SBTTL SUBROUTINE TO LOG AND REPORT TIMEOUTS OF MAP REGISTERS
1098 :*****
1099 :*
1100 :* THIS SUBROUTINE IS USED TO LOG AND REPORT THE FACT THAT A
1101 :* REFERENCE TO A MAPPING REGISTER TIMED OUT ON THE UNIBUS. IT
1102 :* KEEPS A 'LOGICAL AND' AND A 'LOGICAL OR' OF EACH ADDRESS THAT
1103 :* TIMES OUT.
1104 :*
1105 :*****
1106 004410 005227 TIMEOUT:INC (PC)+ ;INCREMENT ONE TIME GATE
1107 004412 177777 TOFLAG: .WORD -1 ;ONE TIME ENTANCE FLAG
1108 004414 001403 BEQ 10$ ;BRANCH IF FLAG IS NOW ZERO
1109 004416 005237 020014 INC $MSGTY ;INDICATE TO APT A FATAL ERROR OCCURED
1110 004422 000000 HALT ;I HAVE ENTERED THIS ROUTINE BEFORE I FINISHED REPORTING THE FIRST ERROR.
1111 ;THE SECOND ENTRY ADDRESS IS ON THE STACK AND THE
1112 ;FIRST ERROR CONDITION IS PROBABLY STILL LOCKED UP.
1113 004424 012637 001326 10$: MOV (KSP)+,OLDPC ;SAVE RETURN ADDRESS
1114 004430 012637 001330 MOV (KSP)+,OLDPS ;SAVE OLD PSW
1115 004434 105737 007774 TSTB CPUTYP ;SEE IF THIS IS AN 11/44
1116 004440 001406 BEQ 1$ ;BRANCH TO CONTINUE IF IT IS
1117 004442 005237 001314 INC PCPUER ;INCREMENT PCPUER TO SHOW A TIMEOUT OCCURED
1118 004446 005737 001312 TST CPUEXP ;SEE IF THERE WAS AN EXPECTED ERROR
1119 004452 001435 BEQ 3$ ;GO REPORT ERROR IF NONE EXPECTED
1120 004454 000442 BR 4$ ;BRANCH TO EXIT IF TIMEOUT WAS EXPECTED
1121 004456 013737 177766 001314 1$: MOV CPUERR,PCPUER ;SAVE CPU ERROR REGISTER
1122 004464 013737 001314 177766 MOV PCPUER,CPUERR ;CLEAR CPU ERROR REGISTER
1123 004472 023737 001314 001312 CMP PCPUER,CPUEXP ;SEE IF EXPECTED CONDITION CAME UP.
1124 004500 001405 BEQ 2$ ;BRANCH IF IT WAS A TIMEOUT
1125 004502 012737 177777 004412 MOV #-1,TOFLAG ;RESET ONE TIME GATE
1126 004510 104001 ERROR +1 ;NOT THE CORRECT CPU TRAP THROUGH 4
1127 004512 000423 BR 4$ ;BRANCH TO EXIT
1128 004514 105737 007774 2$: TSTB CPUTYP ;IS THIS AN 11/24?
1129 004520 001403 BEQ 25$ ;BRANCH IF NOT
1130 004522 005237 001302 INC ERRCNT ;COUNT THIS AS A TIMEOUT
1131 004526 000415 BR 4$ ;GO TO EXIT
1132 004530 022737 000020 001312 25$: CMP #TIMOUT,CPUEXP ;SEE IF A TIMEOUT WAS EXPECTED
1133 004536 001411 BEQ 4$ ;BRANCH TO EXIT THIS ROUTINE IF IT WAS
1134 004540 013746 172356 MOV KIPAR7,-(SP) ;PUT PAR ON STACK FOR ADREXT SUBROUTINE USE
1135 004544 010046 MOV RO,-(SP) ;PUT VIRTUAL ADDRESS ON STACK FOR ADREXT SUBROUTINE USE
1136 004546 004737 005250 3$: JSR PC,ADREXT ;GO SET DATA IN THE 4 WORDS OF ADDROR AND ADRAND
1137 004552 012737 177777 004412 MOV #-1,TOFLAG ;RESET ONE TIME GATE
1138 004560 104201 ERROR +201 ;THE FOLLOWING REGISTERS TIMED OUT WHEN READ
1139 004562 012737 177777 004412 4$: MOV #-1,TOFLAG ;RESET ONE TIME GATE
1140 004570 013746 001330 MOV OLDPS,-(KSP) ;RESTORE OLD PSW
1141 004574 013746 001326 MOV OLDPC,-(KSP) ;PUSH RETURN ADDRESS BACK ON THE STACK
1142 004600 000006 RTT ;RETURN TO THE TEST
    
```

```

1143 .SBTTL CPU TRAP HANDLER ROUTINES
1144 :*****
1145 :*                                     **TRAP HANDLING ROUTINES**
1146 :*****
1147
1148 :*****
1149 :*
1150 :* THIS SUBROUTINE WILL HANDLE ALL CPU TRAPS AND ABORTS, THROUGH
1151 :* 'ERRVEC' (000004). IF THIS SUBROUTINE IS ENTERED BY A SECOND
1152 :* TRAP BEFORE THE FIRST HAS BEEN PROCESSED A HALT IS EXECUTED.
1153 :* IF THE WORD 'CPUEXP' IS ZERO, NO TRAP WAS EXPECTED AND AN
1154 :* UNEXPECTED ERROR MESSAGE IS GIVEN. IF THE WORD 'CPUEXP' IS
1155 :* NOT ZERO THEN THE CPU ERROR REGISTER 'CPUERR' IS COMPARED WITH
1156 :* 'CPUEXP' TO SEE IF THE PROPER CONDITION OCCURRED. 'PCPUER' CAN
1157 :* BE USED AS A FLAG TO INDICATE THAT A TRAP HAS OCCURRED SINCE IT
1158 :* IS LOADED WITH THE ERROR REGISTER IF A TRAP VECTORS HERE
1159 :*
1160 :*****
1161 004602 005227 CPUER: INC (PC)+ ;MAKE FLAG ZERO IF FIRST TIME
1162 004604 177777 CPFLAG: .WORD -1 ;NEGATIVE ONE FOR A FLAG
1163 004606 001403 BEQ 10$ ;BRANCH IF FIRST TIME IN
1164 004610 005237 020014 INC $MSGTY ;INDICATE TO APT A FATAL ERROR OCCURED
1165 004614 000000 HALT ;I HAVE ENTERED THIS ROUTINE BEFORE
1166 :I FINISHED REPORTING THE FIRST ERROR. THE SECOND ENTRY ADDRESS IS ON
1167 :THE STACK, AND THE FIRST ERROR CONDITION IS PROBABLY STILL LOCKED UP.
1168 004616 012637 001326 10$: MOV (KSP)+,OLDPC ;SAVE RETURN ADDRESS IN CASE OF LOOP
1169 004622 012637 001330 MOV (KSP)+,OLDPS ;SAVE OLD PSW IN CASE OF LOOP
1170 004626 013737 177766 001314 MOV CPUERR,PCPUER ;SAVE CPU ERROR REGISTER
1171 004634 013737 001326 001324 MOV OLDPC,BADPC ;SAVE PC+2 AT TIME OF ABORT
1172 004642 005737 001312 TST CPUEXP ;SEE IF ANY CONDITION WAS EXPECTED
1173 004646 001414 BEQ 1$ ;BRANCH IF NO TRAP WAS EXPECTED
1174 004650 105737 007774 TSTB CPUTYP ;SEE IF THIS WAS AN 11/44
1175 004654 001016 BNE 2$ ;BRANCH TO CONTINUE IF AN 11/24
1176 004656 023737 001314 001312 CMP PCPUER,CPUEXP ;SEE IF EXPECTED ERROR OCCURED
1177 004664 001414 BEQ 3$ ;BRANCH IF ERROR CODES MATCH
1178 004666 012737 177777 004604 MOV #-1,CPFLAG ;MAKE FLAG NEGATIVE ONE FOR NEXT TIME
1179 004674 104001 ERROR +1 ;NOT THE CORRECT CPU TRAP THROUGH 4
1180 004676 000407 BR 3$ ;SKIP NEXT INSTRUCTION
1181 004700 012737 177777 004604 1$: MOV #-1,CPFLAG ;MAKE FLAG NEGATIVE ONE FOR NEXT TIME
1182 004706 104002 ERROR +2 ;UNEXPECTED CPU TRAP THROUGH 4
1183 004710 000402 BR 3$ ;SKIP NEXT INSTRUCTION
1184 004712 005237 001302 2$: INC ERRCNT ;INCREMENT ERRCNT TO SHOW AN ERROR FOR 11/24
1185 004716 012737 177777 004604 3$: MOV #-1,CPFLAG ;MAKE FLAG NEGATIVE ONE FOR NEXT TIME
1186 004724 013737 001314 177766 MOV PCPUER,CPUERR ;CLEAR CPU ERROR REGISTER
1187 004732 013746 001330 MOV OLDPS,-(KSP) ;PUSH OLD PSW BACK ON STACK
1188 004736 013746 001326 MOV OLDPC,-(KSP) ;PUSH RETURN ADDRESS BACK ON STACK
1189 004742 000006 RTT ;RETURN FROM INTERRUPT OR ABORT
    
```

```

1190 .SBTTL MEMORY MANAGEMENT TRAPS AND ABORTS HANDLER ROUTINE
1191 :*****
1192 :*
1193 :* THIS ROUTINE WILL HANDLE ALL SPURIOUS MEMORY MANAGEMENT TRAPS
1194 :* AND ABORTS. IT WILL REPORT THE CONDITION OF ALL THE MEMORY
1195 :* MANAGEMENT STATUS REGISTERS, AND THEN RETURN TO THE TEST AND
1196 :* TRY TO CONTINUE RUNNING.
1197 :*
1198 :*****
1199 004744 005227 MMTRAP: INC (PC)+ ;MAKE FLAG ZERO IF FIRST TIME
1200 004746 177777 MMFLAG: .WORD -1 ;FLAG SHOULD BE NEG ONE
1201 004750 001403 BEQ 10$ ;BRANCH IF FIRST TIME INTO ROUTINE
1202 004752 005237 020014 INC $MSGTY ;INDICATE TO APT A FATAL ERROR OCCURED
1203 004756 000000 HALT ;I HAVE ENTERED THIS ROUTINE BEFORE I FINISHED REPORTING THE
1204 :FIRST ERROR. THE SECOND ENTRY ADDRESS IS ON THE STACK AND THE FIRST ERROR
1205 :CONDITION IS PROBABLY STILL LOCKED UP .
1206 004760 011637 001324 10$: MOV (KSP),BADPC ;SAVE PC AT TIME OF ABORT OR TRAP
1207 004764 012637 001326 MOV (KSP)+,OLDPC ;SAVE RETURN ADDRESS IN CASE OF LOOP
1208 004770 012637 001330 MOV (KSP)+,OLDPS ;SAVE OLD PSW IN CASE OF LOOP
1209 004774 013737 177572 001334 MOV MMRO,PMMRO ;SAVE STATUS REGISTER
1210 005002 013737 177574 001336 MOV MMR1,PMMR1 ;SAVE AUTO INC/DEC REGISTER
1211 005010 013737 177576 001340 MOV MMR2,PMMR2 ;SAVE VIRTUAL ADDRESS REGISTER
1212 005016 104003 ERROR +3 ;UNEXPECTED M.M. ABORT OR TRAP
1213 005020 042737 177776 177572 1$: BIC #177776,MMRO ;CLEAR ALL BITS EXCEPT 0
1214 005026 012737 177777 004746 MOV #-1,MMFLAG ;RESTORE A NEGATIVE ONE TO FLAG
1215 005034 013746 001330 MOV OLDPS,-(KSP) ;PUSH OLD PSW ONTO STACK
1216 005040 013746 001326 MOV OLDPC,-(KSP) ;PUSH RETURN ADDRESS ON STACK
1217 005044 000006 RTT ;RETURN TO MAIN PROGRAM
    
```

```

1218 .SBTTL SUBROUTINE TO TEST A LOCATION FOR WRITEABILITY
1219 :*****
1220 :*
1221 :* THIS SUBROUTINE CLEARS A TEST LOCATION, LOADS THE LOCATION USING
1222 :* THE MAP REGISTER, AND DETERMINES IF THE LOCATION WAS LOADED. IF
1223 :* IT WAS, RETURN IS NORMAL TO THE TEST. IF NOT, THE PC ON THE
1224 :* STACK IS UPDATED BY 2 AND THEN A RETURN IS EXECUTED.
1225 :*
1226 :*****
1227 005046 062737 000200 172350 TSTLOC: ADD #200,KIPAR4 ;MAP TO NEXT REGISTER
1228 005054 005237 001172 INC $TMP0 ;INCREMENT REGISTER COUNTER
1229 005060 105737 020034 TSTB $ENV ;TEST APT STATUS
1230 005064 001425 BEQ NEXT ;BRANCH IF NOT UNDER APT
1231 005066 005737 006154 TST FLOATR ;SEE IF BIT 15 OF FLOATR IS SET
1232 005072 100011 BPL 1$ ;BRANCH IF STILL PLUS
1233 005074 022705 020102 CMP #SDDW1,R5 ;SEE IF R5 IS POINTING TO UPPER DDW
1234 005100 001417 BEQ NEXT ;BRANCH IF SO - ALL DONE
1235 005102 012705 020102 MOV #SDDW1,R5 ;MOVE ADDRESS OF DDW1 TO R5 AND
1236 005106 012737 000001 006154 MOV #BIT0,FLOATR ;RESET BIT 0 IN FLOATR
1237 005114 000411 BR NEXT ;BRANCH OVER ASL
1238 005116 005227 1$: INC (PC)+ ;INCREMENT NEXT LOCATION FOR FIRST TIME THROUGH CHECK
1239 005120 177777 FTTHRU: .WORD -1 ;FIRST TIME ENTRANCE FLAG
1240 005122 001004 BNE 1$ ;BRANCH IF NOT FIRST TIME
1241 005124 012737 000001 006154 MOV #BIT0,FLOATR ;MOVE BIT 0 TO LOCATION FLOATR
1242 005132 000402 BR NEXT ;BRANCH OVER THE ASL
1243 005134 006337 006154 1$: ASL FLOATR ;ROTATE THE TEST BIT TO THE LEFT
1244 005140 005037 037776 NEXT: CLR 37776 ;CLEAR TEST LOCATION
1245 005144 005037 001314 CLR PCPUER ;CLEAR ERROR LOCATION
1246 005150 010210 MOV R2,(R0) ;TRY TO LOAD TEST CELL THROUGH MAP
1247 005152 023702 037776 CMP 37776,R2 ;SEE IF TEST LOCATION WAS LOADED
1248 005156 001407 BEQ 2$ ;BRANCH IF IT WAS LOADED
1249 005160 005737 001314 TST PCPUER ;SEE IF A TIMEOUT OCCURED
1250 005164 001402 BEQ 1$ ;BRANCH OVER SPECIAL STACK PUSH IF NOT
1251 005166 013743 001172 MOV $TMP0,-(R3) ;PUSH REGISTER NUMBER THAT TIMED OUT ON SPECIAL STACK
1252 005172 062716 000002 1$: ADD #2,(SP) ;CORRECT PC RETURN FOR LOAD FAILURE INDICATION
1253 005176 000207 2$: RTS PC ;RETURN FROM THIS SUBROUTINE
    
```



```
1254 .SBTTL SUBROUTINE TO LOAD DATAOR AND DATAND
1255 :*****
1256 :*
1257 :* THIS SUBROUTINE ASSUMES THE DATA TO BE ANDED AND ORED HAS BEEN PUT
1258 :* ON THE STACK BEFORE THIS SUBROUTINE WAS CALLED. IT BIT SETS THE
1259 :* DATA ONTO DATAOR, COMPLEMENTS THE DATA AND BIT CLEARS IT ONTO
1260 :* DATAND.
1261 :*
1262 :*****
1263 005200 056637 000002 001244 DATEXT: BIS 2(SP),DATAOR ;SET THE 'OR' PATTERN TO DATAOR
1264 005206 005166 000002 COM 2(SP) ;COMPLIMENT THE DATA
1265 005212 046637 000002 001240 BIC 2(SP),DATAND ;CLEAR THE 'AND' PATTERN TO DATAND
1266 005220 012616 MOV (SP)+,(SP) ;CLEAN THE STACK FOR THE RETURN
1267 005222 000207 RTS PC ;RETURN
```

```
1268 .SBTTL SUBROUTINE TO LOAD PATAOR AND PATAND
1269 :*****
1270 :*
1271 :* THIS SUBROUTINE ASSUMES THE DATA TO BE ANDED AND ORED HAS BEEN PUT
1272 :* ON THE STACK BEFORE THIS SUBROUTINE WAS CALLED. IT BIT SETS THE
1273 :* DATA ONTO PATTOR, COMPLEMENTS THE DATA AND BIT CLEARS IT ONTO
1274 :* PATAND.
1275 :*
1276 :*****
1277 005224 056637 000002 001252 PATEXT: BIS 2(SP),PATTOR ;SET THE 'OR' PATTERN TO PATTOR
1278 005232 005166 000002 COM 2(SP) ;COMPLIMENT THE PATTERN
1279 005236 046637 000002 001250 BIC 2(SP),PATAND ;CLEAR THE 'AND' PATTERN TO PATAND
1280 005244 012616 MOV (SP)+,(SP) ;CLEAN UP STACK FOR RETURN
1281 005246 000207 RTS PC ;RETURN
```

1282
 1283
 1284
 1285
 1286
 1287
 1288
 1289
 1290
 1291
 1292
 1293
 1294
 1295
 1296
 1297
 1298 005250 010546
 1299 005252 016605 000006
 1300 005256 072527 177766
 1301 005262 042705 177700
 1302 005266 010537 001204
 1303 005272 050537 001236
 1304 005276 005105
 1305 005300 040537 001232
 1306 005304 016605 000006
 1307 005310 013766 001204 000006
 1308 005316 042705 176000
 1309 005322 072527 000006
 1310 005326 042766 160000 000004
 1311 005334 060566 000004
 1312 005340 012605
 1313 005342 056637 000002 001234
 1314 005350 005166 000002
 1315 005354 046637 000002 001230
 1316 005362 005166 000002
 1317 005366 012605
 1318 005370 012637 006144
 1319 005374 012637 006146
 1320 005400 010516
 1321 005402 000207

.SBTTL SUBROUTINE TO TAKE PAR AND LOAD 2 WORDS EACH OF ADDROR & ADRAND

THIS SUBROUTINE ASSUMES THE CONTENTS OF THE PAR, AND THE VIRTUAL ADDRESS HAVE BEEN PUT ON THE STACK. IT TAKES THE PAR, SHIFTS IT TO EXPOSE THE UPPER ADDRESS BITS, BIT SETS THEM TO ADDROR+2, COMPLIMENTS THE CONTENTS AND BIT CLEARS ADRAND+2. AFTER RELOADING THE PAR IN R5, IT SHIFTS TO GET THE LOWER 16 BIT EQUIVALENT AND ADDS THE VIRTUAL ADDRESS TO CREATE THE PHYSICAL LOWER 16 BITS. THEN IT BIT SETS THEM TO ADDROR, COMPLIMENTS THE CONTENTS AND BIT CLEARS ADRAND. ANOTHER COMPLIMENT BRINGS THE STATE BACK TO ITS ORIGINAL STATE. THIS SUBROUTINE LEAVES WITH THE LOWER 16 BITS AND THE UPPER 6 BITS ON THE STACK, AND ARE TO BE REMOVED IN THAT ORDER, AND MUST BE REMOVED AFTER RETURN.

```
ADREXT: MOV R5,-(SP) ;SAVE R5
        MOV 6(SP),R5 ;MOVE PAR CONTENTS TO R5 FOR SHIFTING
        ASH #-10,R5 ;SHIFT R5 TO THE RIGHT 10 PLACES
        BIC #177700,R5 ;CLEAR BITS 15 TO 6
        MOV R5,$TMP5 ;MOVE OBTAINED UPPER 6 BITS TO $TMP5 FOR FUTURE TRANSFER
        BIS R5,ADDROR+2 ;SET THE 'OR' PATTERN OF UPPER 6 BITS TO ADDROR+2
        COM R5 ;COMPLIMENT R5
        BIC R5,ADRAND+2 ;CLEAR THE 'AND' PATTERN OF UPPER 6 BITS TO ADRAND+2
        MOV 6(SP),R5 ;PUT PAR CONTENTS BACK IN R5
        MOV $TMP5,6(SP) ;MOVE UPPER 6 BITS OF PHYSICAL ADDRESS ON STACK
        BIC #176000,R5 ;STRIP OFF UPPER 6 BITS OF PAR CONTENTS
        ASH #6,R5 ;SHIFT REMAINING BITS 6 PLACES TO THE LEFT
        BIC #160000,4(SP) ;STRIP OFF PAR PAGE BITS FROM ADDRESS TO FORM OFFSET
        ADD R5,4(SP) ;FORM LOWER 16 BITS OF ADDRESS
        MOV (SP)+,R5 ;RESTORE R5
        BIS 2(SP),ADDROR ;SET THE 'OR' PATTERN TO ADDROR
        COM 2(SP) ;COMPLIMENT THE ADDRESS
        BIC 2(SP),ADRAND ;CLEAR THE 'AND' PATTERN TO ADRAND
        COM 2(SP) ;RETURN ADDRESS TO ITS ORIGINAL STATE
        MOV (SP)+,R5 ;SAVE RETURN ADDRESS
        MOV (SP)+,EADRES ;POP STACK INTO EADRES
        MOV (SP)+,EADRES+2 ;POP STACK INTO EADRES+2
        MOV R5,(SP) ;RESTORE RETURN ADDRESS
        RTS PC ;RETURN
```

```

1322 .SBTTL SUBROUTINE TESTING RELOCATION ADDER
1323 *****
1324 * THE FOLLOWING SUBROUTINE IS USED IN TESTS 13 AND 16, AND USES THE DATA
1325 * BANK FOLLOWING TO EXECUTE THE LOOPS IN THE TEST. R3 IS THE SPECIAL
1326 * 'STACK' POINTER, INITIALIZED AT THE BEGINING TO THE R3STAK DATA BANK.
1327 * THE STACK POINTER IS ADVANCED 3 WORDS FOR EACH PASS. CALL THIS SUB-
1328 * ROUTINE IN THIS MANNER:
1329 * CLR $TMP4 ;CLEAR $TMP4 - USED IN ERROR RETURN
1330 *1$: JSR PC,MAPADD ;GO DO THE TEST
1331 * ERROR +ERRORNUMBER ;RETURN IS HERE FOR ERROR
1332 * BR 1$ ;BRANCH BACK TO CONTINUE TEST
1333 *****
1334 005404 005737 001202 MAPADD: TST $TMP4 ;SEE IF THIS ENTRY WAS AN ERROR
1335 005410 001040 BNE 3$ ;GO CONTINUE TEST
1336 005412 011646 MOV (SP),-(SP) ;MOVE RETURN UP ONE NOTCH
1337 005414 062766 000004 000002 ADD #4,2(SP) ;CREATE ADDRESS ON STACK FOR FINAL RETURN
1338 005422 012703 005530 MOV #R3STAK,R3 ;SET THE SPECIAL STACK POINTER
1339 005426 012704 000013 MOV #13,R4 ;SET THE LOOP COUNTER
1340 005432 012737 005460 001106 MOV #2$,$LPERR ;SET LOOP ON ERROR POINTER TO 2$
1341 005440 005077 173634 CLR @LREGU ;CLEAR UPPER BITS OF MAPPING REG
1342 005444 012377 173626 1$: MOV (R3)+,@LREGL ;LOAD LOWER BITS OF MAPPING REG
1343 005450 013737 001254 172354 MOV LOWEST,KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
1344 005456 012300 MOV (R3)+,R0 ;SELECT PAR6, OFFSET IS AUGEND
1345 005460 011001 2$: MOV (R0),R1 ;READ LOCATION DEFINED BY 4TH WORD IN TABLE
1346 005462 012302 MOV (R3)+,R2 ;MOVE ANTICIPATED DATA TO R2
1347 005464 020102 CMP R1,R2 ;SEE IF THE MAP'S FETCH WAS CORRECT
1348 005466 001415 BEQ 4$ ;BRANCH IF FETCHED DATA MATCHES ADDRESS
1349 005470 162703 000002 SUB #2,R3 ;ANTICIPATE ERROR LOOPING BY UNDOING AUTOINC
1350 005474 010237 006144 MOV R2,EADRES ;PUT ADDRESS IN EADRES FOR ERROR CALL
1351 005500 011646 MOV (SP),-(SP) ;PUT AN EXTRA RETURN ON FOR POSSIBLE ERROR LOOPING
1352 005502 012737 000001 001202 MOV #1,$TMP4 ;SET FLAG SHOWING ERROR CALL WAS CALLED
1353 005510 000207 RTS PC ;EXIT TO ERROR CALL AT TEST
1354 005512 062703 000002 3$: ADD #2,R3 ;RESTORE AUTOINC, ERROR LOOPING NOT DONE
1355 005516 062706 000002 ADD #2,SP ;CLEAN EXTRA RETURN OFF STACK - LOOPING NOT DONE
1356 005522 077430 4$: SOB R4,1$ ;SUBTRACT 1 FROM R4 AND BRANCH IF NOT 0
1357 005524 005726 TST (SP)+ ;EXPOSE NEXT TEST RETURN ADDRESS
1358 005526 000207 RTS PC ;EXIT TO NEXT TEST
1359 ;DATA IN THE R3STAK DATA BANK BELOW IS ARRANGED IN THE FOLLOWING ORDER:
1360 ;>>>NOTE<<<: THE 'OFFSET' COLUMN IS NOT IN THE STACK DUE TO THE DELIMITER
1361 ;(:) BETWEEN THE EXPECTED ADDRESS AND THE OFFSET VALUES
1362 ;
1363 ; :VIRTUAL:PHYSCL:
1364 ; :BASE :ADDRES:ADDRES:OFFSET R4 VALUE
1365 005530 060000 140000 060000 R3STAK: .WORD 060000,140000,060000;000000 R4=13
1366 005536 052524 145252 057776 .WORD 052524,145252,057776;005252 R4=12
1367 005544 045252 152524 057776 .WORD 045252,152524,057776;012524 R4=11
1368 005552 050420 150420 061040 .WORD 050420,150420,061040;010420 R4=10
1369 005560 054630 144210 061040 .WORD 054630,144210,061040;004210 R4=7
1370 005566 044210 154630 061040 .WORD 044210,154630,061040;014630 R4=6
1371 005574 056734 142104 061040 .WORD 056734,142104,061040;002104 R4=5
1372 005602 042104 156734 061040 .WORD 042104,156734,061040;016734 R4=4
1373 005610 057776 141042 061040 .WORD 057776,141042,061040;001042 R4=3
1374 005616 041042 157776 061040 .WORD 041042,157776,061040;017776 R4=2
1375 005624 057776 140002 060000 .WORD 057776,140002,060000;000002 R4=1

```

```

1376 .SBTTL SUBROUTINE TO TEST CARRY PROP OF MAP'S RELOC ADDER
1377 ;*****
1378 ;* THIS SUBROUTINE IS USED BY TESTS 14 AND 17. CODE CALLING THIS SUB-
1379 ;* ROUTINE IS AS FOLLOWS:
1380 ;* CLR $TMP4 ;CLEAR $TMP4 - USED IN ERROR RETURN
1381 ;*LAB: JSR PC,TCPMRA ;GO DO THE TEST
1382 ;* ERROR +211 ;RETURN IS HERE IF AN ERROR
1383 ;* BR LAB ;BRANCH BACK TO CONTINUE TEST
1384 ;*****
1385 005632 005737 001202 TCPMRA: TST $TMP4 ;TEST ERROR FLAG TO SEE IF THIS ENTRY IS FROM ERROR
1386 005636 001072 BNE 4$ ;BRANCH TO CONTINUE TEST IF SO
1387 005640 011646 MOV (SP),-(SP) ;MOVE RETURN ADDRESS UP ONE NOTCH
1388 005642 062766 000004 000002 ADD #4,2(SP) ;CREATE CORRECT FINAL RETURN ADDRESS
1389 005650 012737 177777 006010 MOV #-1,3$+2 ;INITIALIZE FLAG AS NEGATIVE ONE
1390 005656 005077 173416 CLR @LREGU ;CLEAR UPPER 6 BITS OF MAP REG
1391 005662 012777 020000 173406 MOV #20000,@LREGL ;LOAD 4K BASE INTO MAP REGISTER
1392 005670 012701 100100 MOV #100100,R1 ;LOAD BITS TO SELECT PAR 4, OFFSET 100
1393 005674 012700 150000 MOV #150000,R0 ;LOAD BITS TO SELECT PAR 6, OFFSET 2K
1394 005700 012737 000277 172350 MOV #277,KIPAR4 ;START WITH PHYSICAL 6K
1395 005706 013737 001254 172354 MOV LOWEST,KIPAR6 ;LOAD PAR 6 WITH MAP REG'S ADDR
1396 005714 012737 005754 001106 MOV #2$,$LPERR ;SET LOOP ON ERROR POINTER TO 10$
1397 005722 012737 000020 001312 1$: MOV #20,CPUEXP ;EXPECTING A UNIBUS TIME OUT DURING TEST
1398 005730 005037 001314 CLR PCPUER ;CLEAR TIME OUT FLAG
1399 005734 013710 001350 MOV DATA,(R0) ;THIS LOAD WILL TIME OUT WHEN YOU HAVE REACHED THE TOP
1400 ;OF MEMORY IT SELECTS PAR 6 WHICH WILL PUT ADDR <XXX1>0000 ON THE UNIBUS. THE X'S WILL
1401 ;SELECT THE LOWEST USABLE MAPPING REGISTER. THE DEFAULT CASE IS 00010000, SELECTING
1402 ;MAP REGISTER 0.
1403 005740 005737 001314 TST PCPUER ;SEE IF THERE WAS MAIN MEMORY
1404 005744 001020 BNE 3$ ;BRANCH IF NO MAIN MEMORY FROM UNIBUS
1405 005746 012737 000040 001312 MOV #NEXMEM,CPUEXP ;POSSIBLE CACHE NON-EXISTENT MEMORY
1406 005754 011103 2$: MOV (R1),R3 ;READ TEST LOCATION VIA FASTBUS
1407 005756 022737 000040 001314 CMP #NEXMEM,PCPUER ;WAS THIS CACHE NON-EXISTENT MEMORY
1408 005764 001410 BEQ 3$ ;BRANCH IF NON-EXISTENT MEMORY
1409 005766 011002 MOV (R0),R2 ;READ TEST LOCATION VIA UNIBUS MAP
1410 005770 020203 CMP R2,R3 ;COMPARE TEST DATA R2=MAP DATA, R3=FASTBUS DATA
1411 005772 001405 BEQ 3$ ;BRANCH IF IT WAS THE SAME
1412 005774 011646 MOV (SP),-(SP) ;PUT AN EXTRA RETURN ADDRESS ON STACK FOR POSSIBLE LOOP
1413 005776 012737 000001 001202 MOV #1,$TMP4 ;SET FLAG SHOWING AN ERROR RETURN
1414 006004 000207 RTS PC ;RETURN TO THE ERROR
1415 006006 005227 177777 3$: INC #-1 ;INCREMENT ONE TIME ENTRANCE FLAG
1416 006012 001006 BNE 5$ ;BRANCH IF I'VE BEEN HERE BEFORE
1417 006014 013737 172350 001342 MOV KIPAR4,RSIZE ;SAVE UPPER LIMIT OF MEMORY
1418 006022 000402 BR 5$ ;BRANCH OVER ERROR LOOP CORRECTION
1419 006024 062706 000002 4$: ADD #2,SP ;POP EXCESS RETURN ADDRESS OFF STACK
1420 006030 062737 000100 001350 5$: ADD #100,DATA ;CHANGE PATTERN FOR NEXT LOAD
1421 006036 062737 000100 172350 ADD #100,KIPAR4 ;ADD 2K TO PAR4
1422 006044 062777 010000 173224 ADD #10000,@LREGL ;ADD 2K TO MAP REGISTER
1423 006052 001323 BNE 1$ ;BRANCH IF MAP REGISTER NOT ZERO
1424 006054 005277 173220 INC @LREGU ;ADD ONE TO UPPER 6 BITS OF MAP REG
1425 006060 022777 000073 173212 CMP #73,@LREGU ;SEE IF TOP 128K BLOCK HAS BEEN PASSED
1426 006066 103315 BHIS 1$ ;BRANCH IF NOT PAST IT
    
```

1427 006070 005237 001350
1428 006074 042737 177700 001350
1429 006102 052737 000300 001350
1430 006110 005037 001312
1431 006114 005726
1432 006116 000207

INC DATA
BIC #177700,DATA
BIS #300,DATA
CLR CPUEXP
TST (SP)+
RTS PC

M 6

:CHANGE DATA PATTERN FOR NEXT PASS
:CLEAR UPPER 10 BITS OF DATA PATTERN
:START WITH 3XX IN DATA PATTERN
:NO CPU TRAPS EXPECTED FOR AWHILE
:POP STACK EXPOSING FINAL RETURN ADDRESS
:EXIT

SEQ 0077

1433					.SBTTL	DATA TABLES AND ASCII STRINGS USED IN THIS DIAGNOSTIC
1434	006120	006124	021730		DTMSG:	.WORD DTMSG,DFMSG ;POINTER TO DATA VARIABLE ADDRESSES
1435	006124	001264	001266	001270	DTMSG:	.WORD MMRLow,MMRHI,UBRLOW,UBRHI,TESTNO,0 ;DATA VARIABLES TO BE PRINTED
1436	006140	000000			CHARCT:	.WORD 0 ;THIS LOCATION HOLDS CHARACTERS INPUTED DURING THE TYPE ROUTINE
1437	006142	000000			EXTOUT:	.WORD 0 ;THIS LOCATION STORES THE OUTPUT OF THE EXTRACTION ROUTINE
1438	006144	000000	000077		EADRES:	.WORD 0,77 ;LOCATIONS FOR STORING 22 BITS OF THE UBMAR REGISTER ADDRESS
1439	006150	000000	000077		EADRS2:	.WORD 0,77 ;LOCATIONS FOR STORING ANOTHER UBMAR REGISTER ADDRESS
1440	006154	000000			FLOATR:	.WORD 0 ;LOCATION TO HOLD BIT TO FLOAT TO TEST DDW'S STATUS
1441	006156	000000			MASK1:	.WORD 0 ;PRE-LOADED BY THE TEST WITH THE 1ST BIT-MASK
1442	006160	000000			MASK2:	.WORD 0 ;PRE-LOADED BY THE TEST WITH THE 2ND BIT-MASK
1443	006162				SPECST:	.BLKW 40
1444	006262	101	103	103	TOMSG:	.ASCII ?ACCESSING MEMORY THROUGH THE FOLLOWING REGISTER(S)?<CRLF>
1445	006345	103	101	125		.ASCIZ ?CAUSED TIME OUTS (THEY MIGHT BE UNEXPECTED)?<CRLF>
1446	006422	200	123	111	JMPMSG:	.ASCII <CRLF>?SIZE JUMPERS ON UNIBUS MAP ARE NOT IN THEIR DEFAULT?<CRLF>
1447	006507	120	117	123		.ASCII ?POSITION. MAP REGISTERS BETWEEN THE LOWEST AND HIGHEST?<CRLF>
1448	006577	125	123	105		.ASCII ?USEABLE, AND ABOVE THE UNIBUS END NUMBER WILL BE TESTED.?<CRLF>
1449	006670	125	116	111		.ASCII ?UNIBUS MEMORY WILL BE ASSUMED TO BE BETWEEN UNIBUS BEGIN?<CRLF>
1450	006761	101	116	104		.ASCII ?AND END REGISTER NUMBERS IF BIT <5> IS SET IN THE SWR,?
1451	007047	105	116	101		.ASCII ?ENABLING TEST #23 TO EXECUTE.?<CRLF><CRLF>
1452	007106	040	040	114		.ASCII ? LOWEST HIGEST UNIBUS UNIBUS?<CRLF>
1453	007147	040	040	125		.ASCII ? USABLE USABLE BEGIN END?<CRLF>
1454	007206	040	040	040		.ASCIZ ? REG# REG # REG # REG # TEST #?<CRLF>
1455	007260	124	110	105	NOMORE:	.ASCII ?THERE ARE STILL MORE ERRORS BUT WILL NOT BE TYPED. EACH?<CRLF>
1456	007351	105	122	122		.ASCIZ ?ERROR WILL BE COUNTED AND NUMBER PRINTED AT THE END-OF-PASS?<CRLF>
1457	007446	124	110	111	BADCPU:	.ASCII ?THIS DIAGNOSTIC IS DESIGNED FOR AN 11/24 OR 11/44.?<CRLF>
1458	007531	104	111	101		.ASCIZ ?DIAGNOSTIC DOES NOT INTERPRET CPU AS EITHER ONE.?<CRLF>
1459	007613	111	123	040	MRQUES:	.ASCIZ ?IS THERE A MAP REGISTER MODULE IN THIS CPU (Y OR <CR>)?
1460	007702	104	111	101	GMRMOD:	.ASCIZ ?DIAGNOSTIC CHECKS THIS MODULE - INSERT BEFORE RE-RUNNING?<CRLF>
1461	007774	000			CPUTYP:	.BYTE 0 ;LOCATION TO STORE THE CPU TYPE NUMBER

```

1462          .SBTTL PRE-TESTING SETUP
1463          ;*****
1464          ;START OF TEST CODE
1465          ;*****
1466
1467          010000          . =10000          ;START TEST CODE AT ADDRESS 10000 (2K)
1468 010000 010000 012737 000340 177776 START: MOV #340,@#PS          ;;LOCK OUT ALL INTERRUPTS
010006 012706 001100 MOV #SCMTAG,R6          ;;FIRST LOCATION TO BE CLEARED
010012 005026 200$: CLR (R6)+          ;;CLEAR MEMORY LOCATION
010014 022706 001134 CMP #STKS,R6          ;;DONE?
010020 001374 BNE 200$          ;;LOOP BACK IF NO
010022 005037 020022 CLR $PASS          ;;INITIALIZE PASS COUNT
010026 012706 001100 MOV #STACK,SP          ;;SETUP THE STACK POINTER
010032 012737 020140 000020 MOV #SCOPE,@#IOTVEC          ;;IOT VECTOR FOR SCOPE ROUTINE
010040 012737 000340 000022 MOV #340,@#IOTVEC+2          ;;LEVEL 7
010046 012737 020520 000030 MOV #ERROR,@#EMTVEC          ;;EMT VECTOR FOR ERROR ROUTINE
010054 012737 000340 000032 MOV #340,@#EMTVEC+2          ;;LEVEL 7
010062 012737 003724 000034 MOV #TRAP,@#TRAPVEC          ;;TRAP VECTOR FOR TRAP CALLS
010070 012737 000340 000036 MOV #340,@#TRAPVEC+2          ;;LEVEL 7
010076 012737 003766 000024 MOV #SPWRDN,@#PWRVEC          ;;POWER FAILURE VECTOR
010104 012737 000340 000026 MOV #340,@#PWRVEC+2          ;;LEVEL 7
010112 013737 021354 021346 MOV $ENDCT,$EOPCT          ;;SETUP END-OF-PROGRAM COUNTER
010120 005037 001210 CLR $TIMES          ;;INITIALIZE NUMBER OF ITERATIONS
010124 005037 001212 CLR $ESCAPE          ;;CLEAR THE ESCAPE ON ERROR ADDRESS
010130 112737 000001 001113 MOVB #1,$ERMAX          ;;ALLOW ONE ERROR PER TEST
010136 012737 021602 000014 MOV #SRTRN,@#TBITVEC          ;;SET 'T' BIT VECTOR TO SRTRN
010144 012737 000340 000016 MOV #340,@#TBITVEC+2          ;;LEVEL 7
010152 012737 000002 021602 MOV #RTI,$RTRN          ;;SET $RTRN TO A RTI
010160 012737 010206 000010 MOV #65$,@#RESVEC          ;;TRY TO DO A RTT
010166 005046 CLR -(SP)          ;;DUMMY PS
010170 012746 010176 MOV #64$,-(SP)          ;;AND PC
010174 000006 RTT          ;;TRY THE RTT
010176 012737 000006 021602 64$: MOV #RTT,$RTRN          ;;RTT IS LEGAL--SET $RTRN TO A RTT
010204 000402 BR 66$
010206 062706 000010 65$: ADD #10,SP          ;;RTT ILLEGAL--CLEAN OFF THE STACK
010212 012737 000012 000010 66$: MOV #RESVEC+2,@#RESVEC          ;;RESTORE TRAP CATCHER
010220 005037 021610 CLR $TBIT          ;;CLEAR 'T' BIT SWITCH
010224 012737 010602 001104 MOV #TST1+2,$LPADR          ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
010232 012737 010602 001106 MOV #TST1+2,$LPERR          ;;SETUP THE ERROR LOOP ADDRESS
1469 010240 005227 177777 INC #-1          ;;FIRST TIME?
010244 001020 BNE 67$          ;;BRANCH IF NO
010246 022737 021534 000042 CMP #SENDAD,@#42          ;;ACT-11?
010254 001414 BEQ 67$          ;;BRANCH IF YES
010256 104400 010264 TYPE ,68$          ;;TYPE ASCIZ STRING
010262 000411 BR 67$          ;;GET OVER THE ASCIZ
010264 200 103 113 68$: .ASCIZ <CRLF>?CKKUABO UBI MAP?<CRLF>
                                .EVEN
                                67$:
1470 010306 005737 020022 TST $PASS          ;;IS THIS THE FIRST PASS?
1471 010312 001067 BNE 9$          ;;BRANCH IF NOT

```


1472 010314 013746 000004
1473 010320 012737 010410 000004
1474 010326 013746 000006
1475 010332 012737 000340 000006
1476 010340 000007
1477 010342 110037 007774

MOV 4, -(SP) C 7 :SAVE TIMEOUT VECTOR
MOV #2\$,4 :TIMEOUTS TO 104\$
MOV 6, -(SP) :SAVE PS VECTOR
MOV #340,6 :PRIORITY 7
MFPT :DETERMINE PROCESSOR TYPE
MOVB R0,CPUTYP :MOVE THE CPU NUMBER TO CPUTYP

SEQ 0080

1478	010346	105337	007774			DECB	CPUTYP	:MAKE THE 11/44 VALUE ZERO
1479	010352	001005				BNE	1\$:BRANCH TO NEXT TEST IF NOT AN 11/44
1480	010354	104400	021775			TYPE	,CPUMSG	:TYPE THE CPU TYPE HEADER
1481	010360	104400	022056			TYPE	,ELEV44	:TYPE THE 11/44 NUMBER
1482	010364	000421				BR	5\$:CONTINUE
1483	010366	122737	000002	007774	1\$:	CMPB	#2,CPUTYP	:SEE IF THIS IS AN 11/24
1484	010374	001007				BNE	3\$:BRANCH TO FATAL ERROR MESSAGE PRINTING IF NOT
1485	010376	104400	021775			TYPE	,CPUMSG	:TYPE THE CPU TYPE HEADER
1486	010402	104400	022052			TYPE	,ELEV24	:TYPE THE 11/24 NUMBER
1487	010406	000410				BR	5\$:CONTINUE
1488	010410	062706	000004		2\$:	ADD	#4,SP	:CLEAN STACK AFTER TIMEOUT
1489	010414	005237	020014		3\$:	INC	\$MSGTY	:TELL APT THIS IS A FATAL ERROR
1490	010420	104400	007446		4\$:	TYPE	,BADCPU	:TYPE THE BAD CPU MESSAGE
1491	010424	000000				HALT		:FATAL ERROR - THIS DIAGNOSTIC IS WRITTEN
1492								:FOR 11/24 AND 11/44 PROCESSORS ONLY
1493	010426	000774				BR	4\$:DON'T ALLOW CONTINUE
1494	010430	012737	010450	000004	5\$:	MOV	#6\$,4	:TIMEOUTS TO 120\$
1495	010436	022777	177777	170504		CMP	#-1,@SWR	:SEE IF HARDWARE SWITCH REGISTER EXISTS OR CONTAINS -1
1496	010444	001006				BNE	8\$:BRANCH TO CONTINUE IF IT EXISTS AND DOESN'T CONTAIN -1
1497	010446	000402				BR	7\$:BRANCH TO SET UP FOR SOFTWARE SWITCH REGISTER IF SO
1498	010450	062706	000004		6\$:	ADD	#4,SP	:CLEAN STACK AFTER TIMEOUT
1499	010454	012737	000176	001150	7\$:	MOV	#\$SSWR,SWR	:MOVE ADDRESS OF SOFTWARE SWITCH REGISTER TO SWR
1500	010462	012637	000006		8\$:	MOV	(SP)+,6	:RESTORE TIMEOUT PS
1501	010466	012637	000004			MOV	(SP)+,4	:RESTORE TIMEOUT VECTOR
1502	010472	132737	000200	020035	9\$:	BITB	#200,\$ENVM	:IS APT SIZING
1503	010500	001403				BEQ	LOOP	:BRANCH IF NOT
1504	010502	012737	020036	001150		MOV	#\$SWREG,SWR	:USE APT SWITCH REGISTER
1505	010510	012737	004744	000250	LOOP:	MOV	#MMTRAP,MMVEC	:LOAD MEMORY MANAGEMENT TRAP SERVICE ROUTINE ADDRESS
1506	010516	012737	000340	000252		MOV	#340,MMVEC+2	:SET PRIORITY SEVEN
1507	010524	012737	004602	000004		MOV	#CPUER,ERRVEC	:LOAD CPU TRAP SERVICE ROUTINE ADDR
1508	010532	012737	000340	000006		MOV	#340,ERRVEC+2	:SET PRIORITY SEVEN
1509	010540	005037	001312			CLR	CPUEXP	:NOT EXPECTING ANY CPU ERRORS
1510	010544	005037	177572			CLR	MMR0	:START IN 16 BIT MAPPING
1511	010550	005037	172516			CLR	MMR3	:DISABLE MAP AND 22-BIT MAPPING
1512	010554	012700	177777			MOV	#-1,RO	:NEGATIVE ONE USED TO INITIALIZE FLAGS
1513	010560	010037	004604			MOV	RO,CPFLAG	:INITIALIZE FLAGS
1514	010564	010037	004412			MOV	RO,TOFLAG	:INITIALIZE FLAGS
1515	010570	010037	004746			MOV	RO,MMFLAG	:INITIALIZE FLAGS
1516	010574	010037	177766			MOV	RO,CPUERR	:CLEAR CPU ERROR REGISTER

1526

```
.SBTTL TEST 1 - MAP REGISTER RESPONSE TEST
*****
:TEST 1      MAP REGISTER RESPONSE TEST
:
: THIS TEST IS USED TO ENSURE THAT ALL THE UNIBUS MAP REGISTERS
: CAN BE REFERENCED UNDER PROGRAM CONTROL, WITHOUT TIMING OUT.
: THE ADDRESSES OF ANY MAP REGISTERS THAT TIME OUT WILL BE REPORTED
: AND, AT THE END OF THE TEST, A SUMMARY OF THOSE REGISTERS WILL
: BE GIVEN.
*****
```

```
TST1:
010600          000004          SCOPE
010600          004737          JSR      PC,PRETST      ;GO SET UP PRETEST DATA
010602          017662          .WORD    TST2,20$,1    ;DATA USED BY PRETST
1527 010606          011022          010634          000001      MOV      #TIMEOUT,ERRVEC ;LOAD ERRVEC WITH ROUTINE ADDRESS
1528 010614          012737          004410          000004      TSTB    CPUTYP        ;TEST TO SEE WHICH CPU IS RUNNING THIS DIAGNOSTIC
1529 010622          105737          007774          BEQ      5$           ;BRANCH AROUND MAP REGISTER EXISTENCE CHECK IF 11/44
1530 010626          001460          CLR      ERRCNT       ;CLEAR THE ERROR COUNTER
1531 010630          005037          001302          20$:      MOV      @MAPLO,R2     ;READ FIRST MAP REGISTER TO R2
1532 010634          017702          157340          TST      ERRCNT       ;SEE IF THERE WERE ANY ERRORS
1533 010640          005737          001302          BEQ      5$           ;BRANCH TO CHECK THEM ALL IF NONE, THEY ARE IN THIS 11/24
1534 010644          001451          CLR      ERRCNT       ;CLEAR THE ERROR COUNTER
1535 010646          005037          001302          CLRB    SPECST        ;CLEAR THE CHARACTER RECEIVER LOCATION
1536 010652          105037          006162          TSTB    $ENV          ;ARE WE RUNNING UNDER APT
1537 010656          105737          020034          BNE     4$           ;BRANCH IF SO
1538 010662          001036          BNE     4$           ;BRANCH IF SO
1539 010664          104400          007613          TYPE    ,MRQUES       ;ASK USER IF THERE ARE MAP REGISTERS IN THIS 11/24
1540 010666          104400          007613          TSTB    SPECST        ;SEE IF A CHARACTER WAS INPUTED DURING THE PRINTING
1541 010668          105777          170232          1$:      BNE     2$           ;BRANCH TO CHECK INPUT IF SO
1542 010670          105777          170232          TSTB    @TKS          ;SEE IF A CHARACTER IS INPUTED
1543 010672          100375          BPL     1$           ;BRANCH BACK IF NOT
1544 010674          117737          170226          006162      MOVB    @TKB,SPECST   ;MOVE THE CHARACTER TO THE SPECST LOCATION
1545 010676          142737          000200          006162      2$:      BICB    #200,SPECST   ;CLEAR THE PARITY BIT
1546 010678          105777          170214          3$:      TSTB    @TPS          ;SEE IF PRINTER IS READY
1547 010680          100375          BPL     3$           ;BRANCH BACK UNTIL READY
1548 010682          113777          006162          170206      MOVB    SPECST,@TPB   ;PRINT THE CHARACTER
1549 010684          104400          001221          TYPE    ,%CRLF        ;TYPE A <CRLF>
1550 010686          122737          000131          006162      CMPB    #'Y',SPECST   ;SEE IF THIS WAS A 'Y'
1551 010688          001410          BEQ     5$           ;BRANCH AROUND FATAL MESSAGE IF EQUAL TO A 'Y'
1552 010690          104400          007702          TYPE    ,GMRMOD       ;TYPE: 'DIAGNOSTIC CHECKS THIS MODULE - INSERT
1553 010692          000000          HALT    ;FATAL ERROR - WAIT FOR USER ACTION
1554 010694          000726          BR      20$          ;TRY AGAIN - RESTART REQUESTED
1555 010696          005237          020014          4$:      INC     $MSGTY        ;TELL APT THIS IS A FATAL ERROR
1556 010698          000000          HALT    ;HALT - FATAL ERROR
1557 010700          000722          BR      20$          ;TRY AGAIN - RESTART REQUESTED
1558 010702          012700          170200          5$:      MOV     #MAPLO,R0     ;PUT FIRST MAP REGISTER ADDR IN R0
1559 010704          012737          011006          001106      MOV     #6$, $LPERR   ;SET LOOP ON ERROR POINTER TO 6$
1560 011002          012703          000100          MOV     #100,R3       ;TEST ALL MAP REGISTERS
1561 011004          012002          6$:      MOV     (R0)+,R2     ;READ MAP REGISTERS TO R2
1562 011006          077302          SOB     R3,6$        ;DO ALL OF THEM
1563 011008          005737          001302          TST     ERRCNT       ;SEE IF THERE WERE ANY ERRORS
1564 011010          001401          BEQ     TST2         ;GO TO NEXT TEST IF NO ERRORS
1565 011012          104004          ERROR   +4           ;SUMMARY OF MAP REGISTERS THAT TIMED OUT ON READ
```

1580

```
.SBTTL TEST 2 - BIT PATTERN AND CLEAR TEST OF 40 MAP REGISTERS
:*****
:*TEST 2 BIT PATTERN AND CLEAR TEST OF 40 MAP REGISTERS
:*
:* THIS TEST WILL RUN 7 BIT PATTERNS THROUGH BOTH WORDS OF 40 UNIBUS
:* MAP REGISTER LOCATIONS MAPL00 - MAPL37, USING TWO MAJOR PASSES.
:* IT WILL TEST THE LOWER 16 BITS ON THE FIRST PASS, AND THE UPPER
:* 6 BITS ON THE SECOND. THIS TEST WILL MAKE SURE THE REGISTER CAN
:* CLEAR, AND THEN RUN 6 BIT PATTERNS THROUGH EACH UNIBUS MAP REGISTER.
:* IF THE DATA PATTERN 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 EACH OF THE TWO MAJOR
:* PASSES, A SUMMARY OF ALL ERRORS IS GIVEN SO THAT YOU CAN DETERMINE
:* IF THE ERROR IS BIT SENSITIVE OR REGISTER SENSITIVE.
:*****
```

```
011022
011022 000004
011024 004737 017662
011030 011400 011044 000002
1581 011036 012737 004602 000004
1582 011044 005037 001302
1583 011050 012700 170200
1584 011054 005037 006156
1585 011060 012737 000001 006160
1586 011066 012701 000040
1587 011072 012702 000007
1588 011076 012705 021712
1589 011102 011504
1590 011104 043704 006156
1591 011110 010410
1592 011112 011504
1593 011114 043704 006160
1594 011120 020410
1595 011122 001450
1596 011124 011003
1597 011126 011537 001172
1598 011132 043737 006156 001172
1599 011140 013746 172356
1600 011144 010046
1601 011146 004737 005250
1602
1603 011152 011546
1604 011154 004737 005224
1605 011160 011046
1606 011162 004737 005200
1607 011166 010037 006150
1608 011172 012737 011202 001106
1609 011200 000411
1610 011202 011504
1611 011204 043704 006156
```

```
TST2:
SCOPE
JSR PC,PRETST ;GO SET UP PRETEST DATA
.WORD TST3,20$,2 ;DATA USED BY PRETST
MOV #CPUER,ERRVEC ;LOAD CPU TRAP SERVICE ROUTINE ADDRESS
20$: CLR ERRCNT ;CLEAR THE ERROR COUNT LOCATION
MOV #MAPL00,R0 ;MOVE STARTING ADDRESS OF LOWER 16 BITS REGISTER TO R0
CLR MASK1 ;MOVE '0' TO MASK1
MOV #1,MASK2 ;MOVE '1' TO MASK2
1$: MOV #40,R1 ;DO 40 REGISTERS LOOP COUNTER
2$: MOV #7,R2 ;MOVE PATTERN LOOP COUNTER TO R2
MOV #PATRNS,R5 ;MOVE PATTERN START TO R5
3$: MOV (R5),R4 ;MOVE NEXT COUNT PATTERN TO R2
BIC MASK1,R4 ;USE MASK1 PRE-LOADED FOR PROPER LOADING
MOV R4,(R0) ;LOAD MAP REGISTER WITH COUNT PATTERN
MOV (R5),R4 ;RELOAD PATTERN
BIC MASK2,R4 ;USE THE 2ND MASK TO CONSTRUCT EXPECTED VALUE
CMP R4,(R0) ;COMPARE EXPECTED WITH RECEIVED
BEQ 7$ ;BRANCH IF DATA IS OK
4$: MOV (R0),R3 ;READ BAD MAP REGISTER DATA INTO R3
MOV (R5),$TMPO ;MOVE PATTERN TO $TMPO
BIC MASK1,$TMPO ;FORM INPUT PATTERN FOR ERROR PRINTING
MOV KIPAR7,-(SP) ;PUT PAR CONTENTS ON STACK FOR ADREXT SUBROUTINE
MOV R0,-(SP) ;PUT VIRTUAL ADDRESS ON STACK FOR ADREXT SUBROUTINE
JSR PC,ADREXT ;GO SET DATA IN THE 4 WORDS OF ADDROR AND ADRAND
;AND FORM A PHYSICAL 22-BIT ADDRESS
MOV (R5),-(SP) ;PUT PATTERN ON STACK FOR PATEXT SUBROUTINE USE
JSR PC,PATEXT ;GO SET DATA INTO PATTOR AND PATAND
MOV (R0),-(SP) ;PUT DATA ON STACK FOR DATEXT SUBROUTINE
JSR PC,DATEXT ;SUBROUTINE TO LOAD DATAOR AND DATAND
MOV R0,EADRS2 ;MOVE ADDRESS IN R0 TO EADRES FOR ERROR CALL
MOV #5$,$LPERR ;RESET LOOP ON ERROR FOR TIGHT LOOP
BR 6$ ;BRANCH OVER LOOP ON ERROR SECTION
5$: MOV (R5),R4 ;RELOAD PATTERN
BIC MASK1,R4 ;PREPARE PATTERN FOR LOAD
```

1612 011210 010410
1613 011212 011504
1614 011214 043704 006160
1615 011220 020410
1616 011222 001401
1617 011224 104203

68:

MOV R4, (R0)
MOV (R5), R4
BIC MASK2, R4
CMP R4, (R0)
BEQ 65\$
ERROR +203

G 7

:RELOAD MAP REGISTER WITH COUNT PATTERN
:RELOAD PATTERN
:PREPARE PATTERN FOR EXPECTED
:SEE IF PATTERN IS OK NOW
:BRANCH IF SO
:THE BIT PATTERN THROUGH THE MAP REGISTERS FAILED

SEQ 0084

1618	011226	032777	001000	167714	65\$:	BIT	#BIT9,@SWR	;SEE IF LOOP ON ERROR IS SET
1619	011234	001362				BNE	5\$;LOOP BACK IF SO
1620	011236	012737	011102	001106		MOV	#3\$,\$LPERR	;MOVE 3\$ TO LOOP ON ERROR INDICATOR
1621	011244	062705	000002		7\$:	ADD	#2,R5	;UNDO LOOP ON ERROR PREPARATION
1622	011250	005302				DEC	R2	;DECREMENT LOOP COUNTER
1623	011252	001313				BNE	3\$;BRANCH IF STILL MORE PATTERNS FOR THIS REGISTER
1624	011254	062700	000004			ADD	#4,R0	;POINT TO NEXT MAP REGISTER UNDER TEST
1625	011260	005301				DEC	R1	;DECREMENT LOOP COUNTER AND
1626	011262	001303				BNE	2\$;BRANCH IF STILL MORE REGISTERS
1627	011264	005737	001302			TST	ERRCNT	;SEE IF THERE WERE ANY ERRORS
1628	011270	001413				BEQ	9\$;BRANCH IF NO ERRORS
1629	011272	022700	170402			CMP	#MAPH37+4,R0	;SEE IF THIS PASS WAS UPPER 6 BITS
1630	011276	001404				BEQ	8\$;GO SERVICE UPPER 6 BITS ERROR IF SO
1631	011300	005337	001110			DEC	\$ERTTL	;DON'T COUNT ERROR +6 AS ANOTHER ERROR
1632	011304	104006				ERROR	+6	;SUMMARY OF BIT PATTERN FAILURES, LOWER 16 BITS
1633	011306	000407				BR	10\$;GO SET UP DATA FOR TESTING UPPER 6 BITS
1634	011310	005337	001110		8\$:	DEC	\$ERTTL	;DON'T COUNT ERROR +7 AS ANOTHER ERROR
1635	011314	104007				ERROR	+7	;SUMMARY OF BIT PATTERN FAILURES, UPPER 6 BITS
1636	011316	000430				BR	TST3	;GO TO NEXT TEST - LOWER 16 BITS ALREADY TESTED
1637	011320	022700	170402		9\$:	CMP	#MAPH37+4,R0	;SEE IF THIS PASS WAS FOR THE UPPER 6 BITS
1638	011324	001425				BEQ	TST3	;GO TO NEXT TEST IF IT WAS
1639	011326	012700	170202		10\$:	MOV	#MAPH00,R0	;MOVE STARTING ADDRESS OF UPPER 6 BITS REGISTER TO R0
1640	011332	012737	177700	006156		MOV	#177700,MASK1	;MOVE 1ST MASK TO MASK1
1641	011340	012737	177700	006160		MOV	#177700,MASK2	;MOVE 2ND MASK TO MASK2
1642	011346	005037	001252			CLR	PATTOR	;CLEAR PATTOR FOR NEXT PASS
1643	011352	005037	001244			CLR	DATAOR	;CLEAR DATAOR FOR NEXT PASS
1644	011356	012737	177777	001250		MOV	#-1,PATAND	;MOVE -1 TO PATAND FOR NEXT PASS
1645	011364	012737	177777	001240		MOV	#-1,DATAND	;MOVE -1 TO DATAND FOR NEXT PASS
1646	011372	005037	001302			CLR	ERRCNT	;CLEAR ERROR COUNT FOR NEXT PASS
1647	011376	000633				BR	1\$;GO REACCOMPLISH TEST FOR UPPER 6 BITS

1660

```
.SBTTL TEST 3 - DUAL ADDRESS LOADS & READS MAP REG'S
:*****
:*TEST 3          DUAL ADDRESS LOADS & READS MAP REG'S
:*
:* THIS TEST ENSURES THAT ONLY ONE UNIBUS MAP REGISTER IS LOADED
:* DURING A 'MOV #DATA,MAPREG' INSTRUCTION. ALL MAP REGISTERS
:* ARE CLEARED AND ONE REGISTER AT A TIME, STARTING WITH MAPLOO,
:* IS LOADED WITH A -1. THEN, ALL MAP REGISTERS ARE READ,
:* STARTING WITH MAPH37, AND VERIFIED TO BE ZERO. ANY REGISTER
:* THAT IS NOT ZERO AND WHOSE UNIBUS ADDRESS DOES NOT MATCH THAT
:* OF THE REGISTER UNDER TEST IS REPORTED TO BE IN ERROR. AT THE
:* END OF THE TEST A SUMMARY OF ALL DUALED REGISTERS IS GIVEN.
:*
:*****
```

```
TST3:
011400
011400 000004
011402 004737 017662
011406 011710 011470 000003
1661 011414 042737 000001 177572
1662 011422 012700 170200
1663 011426 012702 177700
1664 011432 012701 170400
1665 011436 012737 000077 001172
1666 011444 004737 004352
1667 011450 012703 000100
1668 011454 005037 172516
1669 011460 012704 000100
1670 011464 074237 001172
1671 011470 013710 001172
1672 011474 012701 170400
1673 011500 005741
1674 011502 001424
1675 011504 005011
1676 011506 020100
1677 011510 001421
1678 011512 011137 001200
1679 011516 062701 000002
1680 011522 013746 172356
1681 011526 010146
1682 011530 004737 005250
1683
1684 011534 010046
1685 011536 004737 005200
1686 011542 010037 006150
1687 011546 104202
1688 011550 162701 000002
1689 011554 077427
1690 011556 005020
1691 011560 005303
1692 011562 001336
1693 011564 005737 001302

SCOPE
JSR PC,PRETST ;GO SET UP PRETEST DATA
.WORD TST4,20$,3 ;DATA USED BY PRETST
BIC #1,MMR0 ;TURN OFF MEMORY MANAGEMENT
MOV #MAPLOO,R0 ;LOAD ADDRESS OF MAPLOO IN R0
MOV #177700,R2 ;SET UP XOR DATA
1$: MOV #MAPH37+2,R1 ;PUT ADDRESS OF MAPH37+2 IN R1
MOV #77,$TMP0 ;SET UP TEST PATTERN IN $TMP0
JSR PC,CLRMAP ;CLEAR ALL MAP REGISTERS FOR TEST
MOV #100,R3 ;SET UP LOOP COUNTER FOR TESTING 40 REGISTERS, 2 WORDS EACH
CLR MMR3 ;CLEAR MMR3
2$: MOV #100,R4 ;SET UP LOOP COUNTER FOR CHECKING 40 REGISTERS, 2 WORDS EACH
XOR R2,$TMP0 ;REVERSE BIT STATES OF $TMP0 IN BITS 7 TO 15
20$: MOV $TMP0,(R0) ;LOAD MAP REGISTER UNDER TEST
MOV #MAPH37+2,R1 ;PUT ADDRESS OF MAPH37+2 IN R1
3$: TST -(R1) ;SEE IF MAP REGISTER IS ZERO
BEQ 6$ ;GO SEE IF MORE REGISTERS TO TEST IF = 0
CLR (R1) ;CLEAR THE FAILED LOCATION
CMP R1,R0 ;SEE IF NON-ZERO REGISTER ADDRESS MATCHES ADDRESS LOADED
BEQ 6$ ;GO SEE IF MORE REGISTERS TO TEST IF SO
MOV (R1),$TMP3 ;MOVE FAULTY DATA TO $TMP3
ADD #2,R1 ;PREPARE FOR POSSIBLE ERROR LOOPING
MOV KIPAR7,-(SP) ;PUT PAR ON STACK FOR ADREXT SUBROUTINE USE
MOV R1,-(SP) ;PUT VIRTUAL ADDRESS ON STACK FOR ADREXT SUBROUTINE USE
JSR PC,ADREXT ;GO SET DATA IN THE 4 WORDS OF ADDROR AND ADRAND
;AND FORM A PHYSICAL 22-BIT ADDRESS
MOV R0,-(SP) ;PUT ADDRESS ON STACK FOR DATEXT SUBROUTINE USE
JSR PC,DATEXT ;SUBROUTINE TO LOAD DATAOR AND DATAND
MOV R0,EADRS2 ;MOVE ADDRESS IN R0 TO EADRS2 FOR ERROR CALL
ERROR +202 ;DUAL ADDRESSING ERROR IN THE UNIBUS MAP
5$: SUB #2,R1 ;UNDO ERROR LOOPING PREPARATION - NO LOOPING
6$: SOB R4,3$ ;BRANCH IF MORE REGISTERS TO CHECK
CLR (R0)+ ;CLEAR THE REGISTER JUST TESTED AND POINT TO NEXT REGISTER
DEC R3 ;DECREMENT LOOP COUNTER AND
BNE 2$ ;BRANCH IF MORE REGISTERS TO TEST
TST ERRCNT ;SEE IF THERE WERE ANY ERRORS
```

1694 011570 001403
1695 011572 005337 001110
1696 011576 104005
1697 011600 104420
1698
1699 011602 012700 077406

BEQ RELC22
DEC \$ERTTL
ERROR +5
RELC22: TBITR
MOV #77406,R0

J 7

:GO TO NEXT SECTION IF NO ERRORS
:DON'T COUNT ERROR +5 AS ANOTHER ERROR SEQ 0087
:SUMMARY OF DUAL ADDRESSING ERRORS ON LOADING MAP REGISTERS
:RESTORE THE T BIT TO ITS CONDITION
:BEFORE THE LAST TEST
:MAKE THE KERNEL I-SPACE PAGES ALL


```

1700
1701 011606 012701 172300      MOV      #KIPDR0,R1      ;4K, UPWARD EXPANDABLE, READ/WRITE
1702 011612 010021             MOV      R0,(R1)+       ;MOVE ADDRESS OF KIPDR0 TO R1
1703 011614 010021             MOV      R0,(R1)+       ;KERNEL I-SPACE PAGE 0 KIPDR0
1704 011616 010021             MOV      R0,(R1)+       ;KERNEL I-SPACE PAGE 1 KIPDR1
1705 011620 010021             MOV      R0,(R1)+       ;KERNEL I-SPACE PAGE 2 KIPDR2
1706 011622 010021             MOV      R0,(R1)+       ;KERNEL I-SPACE PAGE 3 KIPDR3
1707 011624 010021             MOV      R0,(R1)+       ;KERNEL I-SPACE PAGE 4 KIPDR4
1708 011626 010021             MOV      R0,(R1)+       ;KERNEL I-SPACE PAGE 5 KIPDR5
1709 011630 010021             MOV      R0,(R1)+       ;KERNEL I-SPACE PAGE 6 KIPDR6
1710 011632 012701 172340      MOV      #KIPAR0,R1     ;MOVE ADDRESS OF KIPAR0 TO R1
1711 011636 005021             CLR      (R1)+          ;MAP KIPAR0 TO PHYSICAL 0
1712 011640 012721 000200      MOV      #200,(R1)+     ;MAP KIPAR1 TO PHYSICAL 4K - 8K
1713 011644 012721 000400      MOV      #400,(R1)+     ;MAP KIPAR2 TO PHYSICAL 8K - 12K
1714 011650 012721 000600      MOV      #600,(R1)+     ;MAP KIPAR3 TO PHYSICAL 12K - 16K
1715 011654 012721 001000      MOV      #1000,(R1)+    ;MAP KIPAR4 TO PHYSICAL 16K - 20K
1716 011660 012721 001200      MOV      #1200,(R1)+    ;MAP KIPAR5 TO PHYSICAL 20K - 24K
1717 011664 012721 170000      MOV      #170000,(R1)+  ;MAP KIPAR6 TO UNIBUS
1718 011670 012721 177600      MOV      #177600,(R1)+  ;MAP KIPAR7 TO I/O PAGE
1719 011674 012737 000001 177572  MOV      #BIT0,MMR0     ;ENABLE FULL 18-BIT MAPPING
1720 011702 012737 000020 172516  MOV      #BIT4,MMR3     ;ENABLE 22-BIT MAPPING
1721                               ;* AT THIS POINT 22-BIT RELOCATION FROM MEMORY MANAGEMENT
1722                               ;* IS ENABLED, WITH THE KIPAR'S MAPPED TO PHYSICAL 0-24K.
1723                               ;* KIPAR6 IS MAPPED TO THE UNIBUS (170000) AND
1724                               ;* KIPAR7 IS MAPPED TO THE I/O PAGE (177600).
    
```

1735

.SBTTL TEST 4 - MAP REGISTER ADDRESS DECODE TEST

*TEST 4 MAP REGISTER ADDRESS DECODE TEST

*:

* THIS TEST TRIES TO VERIFY THAT NONE OF THE INPUTS TO THE ADDRESS
 * DECODER FOR THE UNIBUS MAP REGISTERS IS STUCK TRUE. KIPAR6 IS
 * SET UP TO HOLD 177702 AND R4 HAS THE VIRTUAL ADDRESS TO SELECT
 * MAPLOO, THROUGH KIPAR6. THE TEST THEN CHANGES ONE BIT AT A TIME
 * IN PAR6 SO THAT IT SHOULD NEVER REFERENCE MAPLOO. IF IT DOES, AN
 * ERROR IS REPORTED.

*:

TST4:

011710	011710	000004				SCOPE		
011712	011712	004737	017662			JSR	PC,PRETST	;GO SET UP PRETEST DATA
011716	011716	012130	011724	000004		.WORD	TST5,20\$,4	;DATA USED BY PRETST
1736	011724	012737	000020	001312	20\$:	MOV	#TIMOUT,CPUEXP	;EXPECTING CPU TIME OUT ON UNIBUS
1737	011732	013737	172354	001176		MOV	KIPAR6,\$TMP2	;SAVE KIPAR6 FOR RESTORATION LATER
1738	011740	012737	177702	172354		MOV	#177702,KIPAR6	;PUT MAP REGISTER 0 ADDR IN PAR6
1739	011746	012702	175254			MOV	#175254,R2	;PATTERN FOR TESTING.
1740	011752	010237	170200			MOV	R2,MAPLO	;LOAD MAP REGISTER 0
1741	011756	012700	004000			MOV	#BIT11,R0	;SET BIT 11 TO FLOAT THROUGH PAR6
1742	011762	012704	140000			MOV	#140000,R4	;VIRT.ADDR. TO SELECT PAR6
1743	011766	074037	172354		1\$:	XOR	R0,KIPAR6	;CHANGE A BIT OF MAP REGISTER 0'S ADDR
1744	011772	010046				MOV	R0,-(SP)	;SAVE R0 ON STACK
1745	011774	013746	172354			MOV	KIPAR6,-(SP)	;SAVE KIPAR6 ON STACK
1746	012000	011401				MOV	(R4),R1	;READ LOCATION POINTED TO BY PAR6
1747	012002	020201				CMP	R2,R1	;SEE IF DATA FETCHED MATCHES PATTERN
1748	012004	001035				BNE	3\$;BRANCH IF NOT SAME
1749	012006	005014				CLR	(R4)	;TRY TO CLEAR THIS LOCATION
1750	012010	005737	170200			TST	MAPLO	;SEE IF MAP REGISTER 0 GOT CLEARED
1751	012014	001030				BNE	2\$;BRANCH IF MAP REGISTER NOT ZERO
1752	012016	010237	170200			MOV	R2,MAPLO	;RESTORE MAPLO
1753	012022	013746	172354			MOV	KIPAR6,-(SP)	;PUT PAR ON STACK FOR ADREXT SUBROUTINE USE
1754	012026	010446				MOV	R4,-(SP)	;PUT VIRTUAL ADDRESS ON STACK FOR ADREXT SUBROUTINE USE
1755	012030	004737	005250			JSR	PC,ADREXT	;GO SET DATA IN THE 4 WORDS OF ADDROR AND ADRAND
1756	012034	012737	012044	001106		MOV	#10\$,\$LPERR	;SET LOOP ON ERROR POINTER TO 10\$
1757	012042	000411				BR	11\$;GO CALL ERROR
1758	012044	011401			10\$:	MOV	(R4),R1	;READ LOCATION POINTED TO BY PAR6
1759	012046	020201				CMP	R2,R1	;SEE IF DATA FETCHED MATCHES PATTERN
1760	012050	001013				BNE	3\$;BRANCH IF NOT SAME
1761	012052	005014				CLR	(R4)	;TRY TO CLEAR THIS LOCATION
1762	012054	005737	170200			TST	MAPLO	;SEE IF MAP REGISTER 0 GOT CLEARED
1763	012060	001006				BNE	2\$;BRANCH IF MAP REGISTER NOT ZERO
1764	012062	010237	170200			MOV	R2,MAPLO	;RESTORE MAPLO
1765	012066	104207			11\$:	ERROR	+207	;GOT TO MAPLO WITH ONE BIT DIFFERENT IN ADDRESS FROM 1777020
1766	012070	012737	000077	006146		MOV	#77,EADRES+2	;RESTORE 77 TO EADRES+2
1767	012076	010114			2\$:	MOV	R1,(R4)	;RELOAD THE LOCATION
1768	012100	012637	172354		3\$:	MOV	(SP)+,KIPAR6	;RESTORE KIPAR6
1769	012104	012600				MOV	(SP)+,R0	;RESTORE R0
1770	012106	074037	172354			XOR	R0,KIPAR6	;RESTORE BIT TO ORIGINAL STATUS
1771	012112	006200				ASR	R0	;RIGHT SHIFT ONE PLACE
1772	012114	001324				BNE	1\$;GO CONTINUE TEST IF BIT NOT SHIFTED OUT YET
1773	012116	005037	001312		4\$:	CLR	CPUEXP	;ZERO EXPECTED CPU TRAP FLAG
1774	012122	013737	001176	172354		MOV	\$TMP2,KIPAR6	;RESTORE KIPAR6

1787

.SBTTL TEST 5 - DATA PATH, UNIBUS TO MAIN MEMORY

 *TEST 5 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.

 TST5:

012130	000004					SCOPE		
012130	004737	017662				JSR	PC,PRETST	:GO SET UP PRETEST DATA
012132	012470	012162	000005			.WORD	TST6,20\$,5	:DATA USED BY PRETST
1788 012136	004737	004352				JSR	PC,CLRMAP	:CLEAR ALL MAP REGISTERS
1789 012144	012737	000020	001312			MOV	#TIMOUT,CPUEXP	:TIMEOUTS MIGHT OCCUR IN THIS TEST.
1790 012150	012704	000035				MOV	#35,R4	:DO 35 ACCESSES
1791 012156	012737	170400	172354	20\$:		MOV	#170400,KIPAR6	:START WITH ADDRESS 8K FROM UNIBUS
1792 012162	012737	170400	001206			MOV	#170400,\$TMP6	:MOVE IT TO \$TMP6 ALSO
1793 012170	005037	001314		2\$:		CLR	PCPUER	:CLEAR ERROR CONDITION LOCATION
1794 012176	013700	140000				MOV	140000,R0	:TRY TO READ ADDRESS POINTED TO BY PAR6
1795 012202	005737	001314				TST	PCPUER	:SEE IF READ OF ADDRESS TIMED OUT
1796 012206	001411					BEQ	4\$:BRANCH IF REFERENCE WAS GOOD
1797 012212	062737	000200	172354	3\$:		ADD	#200,KIPAR6	:TRY NEXT 4K BLOCK OF MEMORY
1798 012214	062737	000200	001206			ADD	#200,\$TMP6	:ADD 200 TO \$TMP6 ALSO
1799 012222	077416					SOB	R4,2\$:SUBTRACT 1 FROM R4 AND BRANCH IF NOT DONE
1800 012230	104010					ERROR	+10	:NO UNIBUS ADDRESSES RESPOND
1801 012232	000571					BR	SIZEJ	:BRANCH TO SIZE JUMPER TEST
1802 012234	013737	172354	172352	4\$:		MOV	KIPAR6,KIPAR5	:PUT PAR6 INTO PAR5
1803 012236	013737	001206	001204			MOV	\$TMP6,\$TMP5	:DO SAME TRANSFER
1804 012244	042737	170000	172352			BIC	#170000,KIPAR5	:MAKE PAR5 A NON UNIBUS ADDRESS
1805 012252	042737	170000	001204			BIC	#170000,\$TMP5	:CLEAR SAME BITS
1806 012260	012737	173214	120000			MOV	#173214,120000	:PUT RANDOM NUMBER INTO TEST LOCATION BY FAST BUS
1807 012266	013701	140000				MOV	140000,R1	:READ TEST LOCATION BY UNIBUS
1808 012274	022701	173214				CMP	#173214,R1	:SEE IF DATA WAS READ PROPERLY
1809 012300	001406					BEQ	5\$:DATA OKAY NOW VERIFY DATA PATH
1810 012304	020001					CMP	R0,R1	:SEE IF DATA CHANGED FROM FIRST READ
1811 012306	001004					BNE	5\$:BRANCH AROUND NEXT TRY IF SO
1812 012310	062737	000200	172354			ADD	#200,KIPAR6	:TRY NEXT 4K BLOCK OF MEMORY
1813 012312	077452					SOB	R4,2\$:BRANCH BACK TO TRY NEXT 4K BLOCK
1814 012320	012701	021712		5\$:		MOV	#PATRNS,R1	:LOAD ADDRESS OF BIT PATTERNS IN R1
1815 012322	012702	140000				MOV	#140000,R2	:LOAD VIRTUAL ADDRESS INTO R2
1816 012326	012703	000007				MOV	#7,R3	:DO 7 PATTERNS
1817 012332	011112			6\$:		MOV	(R1),(R2)	:LOAD COUNT INTO TEST LOCATION VIA U.B.
1818 012336	021112					CMP	(R1),(R2)	:COMPARE COUNT WITH DATA READ
1819 012340	001426					BEQ	7\$:BRANCH IF DATA MATCHES
1820 012342	011137	001172				MOV	(R1),\$TMP0	:MOVE EXPECTED PATTERN TO \$TMP0

1821	012350	011237	001174						
1822	012354	011246		MOV	(R2), \$TMP1				
1823	012356	004737	005200	MOV	(R2), -(SP)				
1824	012362	011146		JSR	PC, DATEXT				
1825	012364	004737	005224	MOV	(R1), -(SP)				
1826	012370	012737	012400	JSR	PC, PATEXT				
			001106	MOV	#61\$, \$LPERR				

N 7

```

:MOVE RECEIVED PATTERN TO $TMP1
:PUT DATA ON STACK FOR DATEXT SUBROUTINE USE
:SUBROUTINE TO LOAD DATAOR AND DATAND
:PUT PATTERN ON STACK FOR PATEXT SUBROUTINE USE
:GO SET DATA INTO PATTOR AND PATAND
:SET LOOP ON ERROR POINTER TO 61$

```

SEQ 0091

1827	012376	000403				BR	62\$:BRANCH OVER LOOP ON ERROR SECTION
1828	012400	011112			61\$:	MOV	(R1),(R2)	:LOAD COUNT INTO TEST LOCATION VIA U.B.
1829	012402	021112				CMP	(R1),(R2)	:COMPARE COUNT WITH DATA READ
1830	012404	001401				BEQ	63\$:BRANCH IF OK NOW
1831	012406	104204			62\$:	ERROR	+204	:REPORT ERROR(S) ON UNIBUS DATA PATH
1832	012410	032777	001000	166532	63\$:	BIT	#BIT9,@SWR	:SEE IF LOOP ON ERROR IS SET
1833	012416	001370				BNE	61\$:BRANCH BACK IF SO
1834	012420	062701	000002		7\$:	ADD	#2,R1	:MOVE TO NEXT PATTERN
1835	012424	077334				GOB	R3,6\$:DECREMENT LOOP COUNTER AND BRANCH IF NOT DONE
1836	012426	005737	001302			TST	ERRCNT	:WERE THERE ANY ERRORS ON THIS TEST
1837	012432	001403				BEQ	8\$:BRANCH IF NO ERRORS ON THIS TEST
1838	012434	005337	001110			DEC	\$ERTTL	:DON'T COUNT ERROR +11 AS ANOTHER ERROR
1839	012440	104011				ERROR	+11	:SUMMARY OF ERRORS ON THE UNIBUS DATA PATH
1840	012442	005037	001312		8\$:	CLR	CPUEXP	:ZERO EXPECTED CPU TRAP CONDITION
1841	012446	023737	001204	172352		CMP	\$TMP5,KIPAR5	:MAKE SURE PAR5 CONTAINS EXPECTED CONTENTS
1842	012454	001405				BEQ	TST6	:;BRANCH IF OK
1843	012456	104030				ERROR	+30	:KIPAR5 NOT LOADED PROPERLY - SKIPPING NEXT TEST
1844	012460	105237	001100			INCB	\$TSTNM	:INCREMENT TEST POINTER TO SHOW SKIPPED TEST
1845	012464	000137	012604			JMP	TST7	:JUMP OVER NEXT TEST - DEPENDS ON PAR5 SET PROPERLY

1859

.SBTTL TEST 6 - MAP DOESN'T RELOCATE IF NOT ENABLED

*TEST 6 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.

*

TST6:

012470	000004					SCOPE		
012470	004737	017662				JSR	PC,PRETST	:GO SET UP PRETEST DATA
012472	012604	012556	000006			.WORD	TST7,20\$,6	:DATA USED BY PRETST
1860	012504	012737	000020	001312		MOV	#TIMOUT,CPUEXP	:MIGHT TIME OUT OVER UNIBUS
1861	012512	052737	000001	177572		BIS	#BIT0,MMR0	:TURN MEMORY MANAGEMENT BACK ON
1862	012520	013700	172354			MOV	KIPAR6,R0	:PUT UNIBUS ADDRESS OF MAP REGISTER IN R0
1863	012524	072027	177773			ASH	#-5,R0	:RIGHT SHIFT R0 5 PLACES
1864	012530	042700	177400			BIC	#177400,R0	:CLEAR UPPER BYTE
1865	012534	042737	177000	172352		BIC	#177000,KIPAR5	:MAKE KIPAR5 ACCESS THE FAST BUS
1866	012542	012720	021042			MOV	#021042,(R0)+	:SET BOTTOM BIT IN EACH ALU
1867	012546	012710	000042			MOV	#42,(R0)	:SET BOTTOM BIT IN EACH ALU
1868	012552	005037	120000			CLR	120000	:CLEAR TEST LOCATION VIA FAST BUS
1869	012556	012737	043207	140000	20\$:	MOV	#43207,140000	:LOAD TEST LOCATION VIA UNIBUS
1870								:THIS LOAD SHOULD NOT BE RELOCATED
1871								:BY THE UNIBUS MAP, SINCE BIT05 OF MMR3 IS CLEAR.
1872	012564	013703	120000			MOV	120000,R3	:READ TEST LOCATION VIA FAST BUS
1873	012570	022703	043207			CMP	#43207,R3	:SEE IF DATA MATCHES
1874	012574	001401				BEQ	1\$:BRANCH IF DATA GOOD
1875	012576	104012				ERROR	+12	:MAP RELOCATED WHEN NOT ENABLED
1876	012600	005037	001312		1\$:	CLR	CPUEXP	:ZERO EXPECTED CPU TRAP CONDITION

1888

```
.SBTTL TEST 7 - SIZE JUMPER LOCATION TEST
:*****
:*TEST 7          SIZE JUMPER LOCATION TEST
:*
:* 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 INFORMATIONAL MESSAGE IS GIVEN.
:*
:* >>>>>>>>NOTE<<<<<<<<<
:* THIS IS THE FIRST TEST IN WHICH THE UNIBUS MAP IS TURNED ON.
```

```
012604
012604 000004
012606 004737 017662
012612 013440 012620 000007
1889 012620
1890 012620 012737 000020 001312
1891 012626 012705 020100
1892 012632 012700 170200
1893 012636 012701 000040
1894 012642 012702 006162
1895 012646 012720 020000
1896 012652 005020
1897 012654 005022
1898 012656 077105
1899 012660 012703 006162
1900 012664 052737 000040 172516
1901 012672 052737 000001 177572
1902 012700 012700 117776
1903 012704 012737 167600 172350
1904 012712 012702 125252
1905 012716 012737 177777 001172
1906 012724 012737 177777 005120
1907 012732 004737 005046
1908 012736 000412
1909 012740 004737 021616
1910 012744 104214
1911 012746 022737 177400 172350
1912 012754 001366
1913 012756 104013
1914 012760 000137 010000
1915 012764 004737 021654
1916 012770 104213
1917 012772 013737 001172 001264
1918 013000 013737 172350 001254
1919 013006 022737 170000 001254
1920 013014 001436
1921 013016 005037 001270
1922 013022 012737 170000 001260
```

```
TST7:
SCOPE
JSR PC,PRETST ;GO SET UP PRETEST DATA
.WORD TST10,20$,7 ;DATA USED BY PRETST
20$:
SIZEJ: MOV #TIMOUT,CPUEXP ;EXPECTING CPU TIME OUT ON UNIBUS
MOV #$DDW0,R5 ;PUT ADDRESS OF $DDW0 IN R5
MOV #MAPLO,R0 ;LOAD ADDRESS OF FIRST MAP REGISTER IN R0
MOV #40,R1 ;DO ALL 40 REGISTERS
MOV #SPECST,R2 ;SET R2 TO BEGINING OF SPECIAL STACK
1$: MOV #20000,(R0)+ ;LOAD 4K INTO LOWER 16 BITS AND
CLR (R0)+ ;CLEAR THE UPPER 6 BITS
CLR (R2)+ ;CLEAR THE SPECIAL STACK LOCATION
SOB R1,1$ ;BRANCH IF THERE ARE MORE TO LOAD
MOV #SPECST,R3 ;RESET SPECIAL STACK POINTER
BIS #BIT5,MMR3 ;TURN ON MAP RELOCATION
BIS #BIT0,MMR0 ;MAKE SURE MEMORY MANAGEMENT IS ON
MOV #117776,R0 ;THIS WILL BE USED TO SELECT PAR 4; ADDRESS 17776
MOV #167600,KIPAR4 ;LOAD MAP REGISTER 0 -200 IN KIPAR4
MOV #125252,R2 ;CONSTANT TO LOAD INTO LOCATION 17776
MOV #-1,$TMP0 ;MOVE -1 TO MAP REGISTER POINTER
MOV #-1,FTTHRU ;INITIALIZE ONE TIME ENTRANCE FLAG IN SUBROUTINE
2$: JSR PC,$TSTLOC ;GO TO SUBROUTINE TO SEE IF LOCATION RESPONDS
BR 3$ ;RETURN IS HERE IF LOWEST FOUND
JSR PC,$DABLD ;GO SEE IF REGISTER SHOULD BE DISABLED
ERROR +214 ;MAP REGISTER DISABLED WHEN SHOULD BE ENABLED
CMP #177400,KIPAR4 ;SEE IF WE ARE POINTING JUST BELOW THE I/O PAGE
BNE 2$ ;GO TEST NEXT MAP REGISTER IF NOT
ERROR +13 ;FATAL ERROR, RESTARTING PROGRAM
JMP START ;JUMP TO RESTART PROGRAM
3$: JSR PC,$ENABLD ;GO CHECK TO SEE IF REGISTER SHOULD BE ENABLED
ERROR +213 ;MAP REGISTER ENABLED WHEN SHOULD BE DISABLED
MOV $TMP0,MMRLOW ;MOVE REGISTER NUMBER FOUND USEABLE TO MMRLOW
MOV KIPAR4,LOWEST ;MOVE LOWEST USEABLE REGISTER TO LOWEST
CMP #170000,LOWEST ;SEE IF LOWEST REGISTER FOUND WAS THE LOWEST
BEQ 5$ ;BRANCH AROUND SETUP IF IT WAS
CLR UBRLow ;MAP REGISTER 0 IS LOWEST FOR UNIBUS MEMORY
MOV #170000,UBMLow ;MOVE PAR VALUE OF UB MEMORY TO UBMLow
```

1923 013030 013737 001172 001272
1924 013036 005337 001272
1925 013042 013737 172350 001262
1926 013050 162737 000200 001262
1927 013056 012737 177400 001256
1928 013064 012737 000031 001266

E 8

MOV	\$TMPO,UBRHI	:MOVE REGISTER NUMBER FOUND USEABLE TO UBRHI	
DEC	UBRHI	:POINT IT AT HIGHEST UB MEMORY MAP REGISTER	SEQ 0095
MOV	KIPAR4,UBMHI	:MOVE KIPAR4 TO UNIBUS MAP HIGHEST AND	
SUB	#200,UBMHI	:SUBTRACT 200 FROM IT TO POINT TO LAST USEABLE UBMEM PAGE	
MOV	#177400,HIGEST	:MOVE HIGHEST REGISTER TO HIGEST AND	
MOV	#31,MMRHI	:POINT TO LAST USEABLE MAP REGISTER	

1929	013072	000474				BR	11\$:GO TYPE MESSAGE OF NON-DEFAULT INFORMATION
1930	013074	022737	177400	172350	4\$:	CMP	#177400,KIPAR4		:SEE IF ALL MAP REGISTERS HAVE BEEN TRIED
1931	013102	001411				BEQ	6\$:BRANCH IF ALL ARE DONE
1932	013104	004737	021654			JSR	PC,ENABLD		:GO SEE IF MAP REGISTER SHOULD BE ENABLED
1933	013110	104213				ERROR	+213		:MAP REGISTER ENABLED WHEN SHOULD BE DISABLED
1934	013112	004737	005046		5\$:	JSR	PC,TSTLOC		:GO TO SUBROUTINE TO SEE IF IT RESPONDS
1935	013116	000766				BR	4\$:RETURN IS HERE IF IT WAS LOADED
1936	013120	004737	021616			JSR	PC,DSABLD		:RETURN IS HERE IF NOT LOADED - GO SEE IF
1937									:REGISTER SHOULD BE DISABLED
1938	013124	104214				ERROR	+214		:MAP REGISTER DISABLED WHEN SHOULD BE ENABLED
1939	013126	013737	001172	001266	6\$:	MOV	\$TMP0,MMRHI		:MOVE REGISTER FOUND TO MMRHI
1940	013134	005337	001266			DEC	MMRHI		:DECREMENT THIS VALUE - IT IS ONE TOO MANY
1941	013140	013737	172350	001256		MOV	KIPAR4,HIGEST		:MOVE FIRST UNUSABLE REGISTER TO HIGEST
1942	013146	023727	001256	177400		CMP	HIGEST,#177400		:SEE IF UPPER JUMPER IS DEFAULT.
1943	013154	001505				BEQ	16\$:BRANCH AROUND MESSAGE TYPEOUT IF IT IS DEFAULT
1944	013156	013737	001256	001260	7\$:	MOV	HIGEST,UBMLOW		:MOVE UPPER LIMIT TO UBMLOW LOCATION
1945	013164	062737	000200	001260		ADD	#200,UBMLOW		:POINT UBMLOW TO FIRST USABLE UNIBUS MEM PAGE
1946	013172	013737	001172	001270		MOV	\$TMP0,UBRLOW		:MOVE REGISTER NUMBER TO UBRLOW
1947	013200	000407				BR	9\$:BRANCH OVER CHECK FOR NON-EXISTENT LOCATION
1948	013202	022737	177400	172350	8\$:	CMP	#177400,KIPAR4		:SEE IF WE ARE POINTING JUST BELOW THE I/O PAGE
1949	013210	001407				BEQ	10\$:GO INITIALIZE UBMHI IF WE ARE
1950	013212	004737	021616			JSR	PC,DSABLD		:GO SEE IF LOCATION SHOULD BE DISABLED
1951	013216	104214				ERROR	+214		:MAP REGISTER DISABLED WHEN SHOULD BE ENABLED
1952	013220	004737	005046		9\$:	JSR	PC,TSTLOC		:GO TO SUBROUTINE TO SEE IF IT DOESN'T RESPOND
1953	013224	000401				BR	10\$:RETURN IS HERE IF IT WAS LOADED - GO INITIALIZE UBMHI
1954	013226	000765				BR	8\$:RETURN IS HERE IF NOT - GO BACK FOR ANOTHER TRY
1955	013230				10\$:			:AT THIS POINT, KIPAR4	POINTS JUST ABOVE HIGHEST ADDRESS OF UNIBUS MEMORY
1956	013230	004737	021654			JSR	PC,ENABLD		:GO SEE IF MAP REGISTER SHOULD BE ENABLED
1957	013234	104213				ERROR	+213		:MAP REGISTER ENABLED WHEN SHOULD BE DISABLED
1958	013236	013737	172350	001274		MOV	KIPAR4,BUPWIN		:MOVE THIS VALUE TO 'B'EGINING 'UP'PER 'WIN'DOW
1959	013244	013737	172350	001262		MOV	KIPAR4,UBMHI		:MOVE THIS TO UBMHI ALSO
1960	013252	013737	001172	001272		MOV	\$TMP0,UBRHI		:MOVE MAP REGISTER POINTER TO UBRHI
1961	013260	005337	001272			DEC	UBRHI		:DECREMENT THIS VALUE - IT IS ONE TOO MANY
1962	013264	005737	020022		11\$:	TST	\$PASS		:SEE IF THIS IS FIRST PASS
1963	013270	001037				BNE	16\$:BRANCH TO NEXT SECTION IF FIRST PASS
1964	013272	104400	006422			TYPE	,JMPMSG		:TYPE SIZE JUMPERS NOT IN DEFAULT - FOR INFO ONLY
1965	013276	012700	006120			MOV	#DTMS,R0		:SET UP MESSAGE POINTER
1966	013302	012746	013316			MOV	#12\$,-(SP)		:PUSH RETURN ON THE STACK
1967	013306	010046				MOV	R0,-(SP)		:PUSH R0 ON THE STACK
1968	013310	010146				MOV	R1,-(SP)		:PUSH R1 ON THE STACK
1969	013312	000137	002214			JMP	TYPDAT		:GO TYPE THE DATA
1970	013316	104400	001221		12\$:	TYPE	,\$CRLF		:TYPE A <CRLF>
1971	013322	104400	001221			TYPE	,\$CRLF		:TYPE ONE MORE <CRLF>
1972	013326	022703	006162			CMP	#SPECST,R3		:SEE IF ANY TIMEOUTS OCCURED
1973	013332	001416				BEQ	16\$:BRANCH TO NEXT SECTION IF NONE
1974	013334	104400	006262			TYPE	,TOMSG		:TYPE THE TIMEOUTS MESSAGE
1975	013340	012703	006162			MOV	#SPECST,R3		:RESET R3
1976	013344	014346			13\$:	MOV	-(R3),-(SP)		:PUSH THE REGISTER # ONTO THE STACK IN ORDER IT WAS PUSHED
1977	013346	104410			14\$:	TYPDS			:TYPE THE NUMBER IN DECIMAL, LEADING ZEROS SUPPRESSED
1978	013350	104400	001221			TYPE	,\$CRLF		:TYPE A <CRLF>
1979	013354	005743				TST	-(R3)		:SEE IF THERE IS ANOTHER REGISTER NUMBER TO PRINT

1980 013356 001402
1981 013360 011346
1982 013362 000771
1983 013364 104400 001221
1984
1985

15\$:
::
:*

G 8
BEQ 15\$;BRANCH AROUND SETUP IF NONE
MOV (R3),-(SP) ;PUSH THIS NUMBER ON THE STACK
BR 14\$;BRANCH BACK TO PRINT IT
TYPE ,SCLF ;TYPE ONE MORE <CRLF>
SETUP POINTERS TO THE LOWEST AND THE HIGHEST USABLE

SEQ 0097

```
1986  
1987  
1988 013370 005037 001312  
1989 013374 010546  
1990 013376 013705 001254  
1991 013402 042705 170000  
1992 013406 072527 177773  
1993 013412 052705 170200  
1994 013416 010537 001276  
1995 013422 012605  
1996 013424 013737 001276 001300  
1997 013432 062737 000002 001300
```

;* MAPPING REGISTERS TO CONTINUE WITH TEST.
::
16s: CLR CPUEXP ;NO CPU TRAPS EXPECTED IN THIS SECTION
MOV R5,-(SP) ;SAVE R5
MOV LOWEST,R5 ;MOVE PAR DATA TO R5 FOR CONVERSION
BIC #170000,R5 ;CLEAR BITS 15 TO 12
ASH #-5,R5 ;SHIFT INDICATOR BITS TO THE RIGHT 5 PLACES
BIS #170200,R5 ;FORM ADDRESS
MOV R5,LREGL ;SAVE RESULTS
MOV (SP)+,R5 ;RESTORE R5
MOV LREGL,LREGU ;MOVE ADDRESS TO LREGU
ADD #2,LREGU ;POINT TO UPPER 6 BITS OF MAP REG

2016

.SBTTL TEST 10 - ENSURE THAT THERE IS NO DUAL MAPPING

*TEST 10 ENSURE THAT THERE IS NO DUAL MAPPING

*

* THIS TEST VERIFIES THAT THERE IS NO DUAL MAPPING. IT CLEARS
 * ALL THE MAP REGISTERS EXCEPT THE ONE UNDER TEST, AND LOADS
 * THAT ONE WITH 00040000. IF MAP RELOCATION IS ENABLED (AND
 * IN THIS TEST IT IS), SUBROUTINE CLRMAP CLEARS ALL BUT THE
 * LOWER 16 BITS OF MAPL1, AND LOADS 20000 THERE. THIS IS SO
 * THAT APT, IF CONTROLLING THIS DIAGNOSTIC, CAN STILL EXAMINE
 * THE PROPER LOCATIONS. THE TEST THEN USES A VIRTUAL ADDRESS
 * TO SELECT THAT MAP REGISTER AND ADD 17776, SO THAT IT SHOULD
 * REFERENCE ADDRESS 00057776 (00037776 IF MAPL1 CONTAINS 20000
 * AS PER CONDITIONS DESCRIBED ABOVE). 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.

*

TST10:

013440	000004				SCOPE		
013440	004737	017662			JSR	PC,PRETST	:GO SET UP PRETEST DATA
013442	014130	013606	000010		.WORD	TST11,20\$,10	:DATA USED BY PRETST
2017 013446	005037	172516			CLR	MMR3	:CLEAR MMR3
2018 013454	052737	000060	172516		BIS	#60,MMR3	:ENABLE 22-BIT ADDRESSING AND MAP RELOCATION
2019 013460	004737	004352			JSR	PC,CLRMAP	:CLEAR ALL MAP REGISTERS
2020 013466	042737	000001	177572		BIC	#1,MMR0	:TURN OFF MEMORY MANAGEMENT
2021 013472	005037	001172			CLR	\$TMPO	:\$TMPO IS USED AS A FLAG IN THIS TEST
2022 013500	012703	117776			MOV	#117776,R3	:SELECT P.A.R. 4 OFFSET OF 17776
2023 013504	013702	001276			MOV	LREGL,R2	:PUT ADDRESS OF LOWEST USABLE MAP REGISTER IN R2
2024 013510	013700	001254			MOV	LOWEST,R0	:LOAD PAR POINTING TO MAP REGISTER UNDER TEST IN R0
2025 013514	052737	000001	177572		BIS	#1,MMR0	:MAKE SURE MEMORY MANAGEMENT IS ON
2026 013520	005037	001302			CLR	ERRCNT	:CLEAR THE ERROR COUNT FOR ERROR 202 BELOW
2027 013526	005037	001302			MOV	LOWEST,KIPAR4	:PAR OF LOWEST USEABLE MAP REGISTER IS LOADED IN KIPAR4
2028 013532	013737	001254	172350	1\$:	CMP	#MAPL01,R2	:SEE IF WE ARE POINTING AT MAPL1
2029 013540	022702	170204			BNE	13\$:BRANCH IF NOT
2030 013544	001003				CMP	#20000,(R2)	:SEE IF IT CONTAINS 20000 (FOR APT USE)
2031 013546	022712	020000			BEQ	14\$:BRANCH IF SO
2032 013552	001405				MOV	#40000,(R2)	:LOAD MAP REGISTER UNDER TEST WITH 8K BASE
2033 013554	012712	040000		13\$:	MOV	R2,57776	:LOAD TEST LOCATION WITH THE ADDRESS
2034 013560	010237	057776					:OF THE MAP REGISTER UNDER TEST
2035 013564	000402				BR	15\$:BRANCH OVER LOCATION SETUP
2036 013566	010237	037776		14\$:	MOV	R2,37776	:LOAD TEST LOCATION WITH THE ADDRESS
2037							:OF THE MAP REGISTER UNDER TEST
2038 013572	162737	000200	172350	15\$:	SUB	#200,KIPAR4	:PREPARE KIPAR4 FOR FIRST 'ADD 200'
2039 013576	062737	000200	172350	21\$:	ADD	#200,KIPAR4	:TRY NEXT MAP REGISTER
2040 013600	005037	001314		20\$:	CLR	PCPUER	:CLEAR THE ERROR RECEIVER
2041 013606	011304				MOV	(R3),R4	:READ THROUGH THE MAP REGISTER
2042 013612	005737	001314			TST	PCPUER	:SEE IF THERE WAS AN ERROR
2043 013618	001023				BNE	4\$:BRANCH AROUND DATA TEST IF SO

2044 013622 020402
2045 013624 001021
2046 013626 020037 172350
2047 013632 001414
2048 013634 011337 001200
2049 013640 013746 172356

CMP R4,R2
BNE 4\$
CMP R0,KIPAR4
BEQ 3\$
MOV (R3),\$TMP3
MOV KIPAR7,-(SP)

J 8

;SEE IF CORRECT DATA WAS FETCHED
;BRANCH IF NO MATCH
;SEE IF MAP REGISTERS ARE THE SAME
;BRANCH IF CORRECT MAP REGISTER WAS USED
;SAVE CONTENTS FOR ERROR PRINTING
;PUT PAR ON STACK FOR SUBROUTINE ADREXT USE

SEQ 0100

2050	013644	010246				MOV	R2,-(SP)	:PUT ADDRESS OF REGISTER ON STACK FOR ADREXT USE
2051	013646	004737	005250			JSR	PC,ADREXT	:GO SET DATA IN THE 4 WORDS OF ADDROR AND ADRAND
2052	013652	010246				MOV	R2,-(SP)	:PUT ADDRESS OF REGISTER ON STACK FOR DATEXT USE
2053	013654	004737	005200			JSR	PC,DATEXT	:GO SET DATA IN DATAOR AND DATAND
2054	013660	104210				ERROR	+210	:DUAL MAPPING ERROR IN THE UNIBUS MAP
2055	013662	000402				BR	4\$:BRANCH AROUND ADDRESS MATCH SETTING
2056	013664	005237	001172		3\$:	INC	\$TMPO	:SET FLAG WHEN ADDRESSES MATCH
2057	013670	022737	177400	172350	4\$:	CMP	#177400,KIPAR4	:SEE IF LAST REGISTER HAS BEEN TRIED
2058	013676	001423				BEQ	6\$:BRANCH TO CONTINUE IF SO
2059	013700	023737	001256	172350		CMP	HIGEST,KIPAR4	:SEE IF ALL HAVE BEEN TRIED
2060	013706	001334				BNE	21\$:BRANCH IF STILL MORE TO TRY
2061	013710	062737	000200	172350	5\$:	ADD	#200,KIPAR4	:MAP TO NEXT MAP REGISTER
2062	013716	022737	177400	172350		CMP	#177400,KIPAR4	:SEE IF WE ARE AT THE TOP
2063	013724	001410				BEQ	6\$:BRANCH TO CONTINUE IF SO
2064	013726	023737	001274	172350		CMP	BUPWIN,KIPAR4	:SEE IF WE ARE POINTING TO THE UPPER WINDOW START
2065	013734	001365				BNE	5\$:BRANCH BACK FOR ANOTHER INCREMENTING SET IF NOT
2066	013736	162737	000200	172350		SUB	#200,KIPAR4	:PREPARE FOR THE 'ADD 200'
2067	013744	000720				BR	20\$:GO BACK FOR A NEW TRY
2068	013746	005737	001172		6\$:	TST	\$TMPO	:SEE THAT THERE WAS A SUCCESSFUL MATCH
2069	013752	001017				BNE	7\$:BRANCH IF THERE WAS
2070	013754	010005				MOV	R0,R5	:MOVE R0 TO R5 FOR PREPARATION AND SHIFTING
2071	013756	042705	170000			BIC	#170000,R5	:CLEAR THE UPPER 4 BITS
2072	013762	072527	177773			ASH	#-5,R5	:SHIFT RIGHT 5 PLACES
2073	013766	062705	170200			ADD	#170200,R5	:FORM LOWER 16 BITS OF PHYSICAL ADDRESS OF MAP REGISTER
2074	013772	010537	006144			MOV	R5,EADRES	:MOVE THE OBTAINED ADDRESS TO THE ERROR LOCATION
2075	013776	013746	172356			MOV	KIPAR7,-(SP)	:PUT PAR ON STACK FOR SUBROUTINE ADREXT USE
2076	014002	010246				MOV	R2,-(SP)	:PUT ADDRESS OF REGISTER ON STACK FOR ADREXT USE
2077	014004	004737	005250			JSR	PC,ADREXT	:GO SET DATA IN THE 4 WORDS OF ADDROR AND ADRAND
2078	014010	104210				ERROR	+210	:DUAL MAPPING ERROR IN THE UNIBUS MAP
2079	014012	005037	001172		7\$:	CLR	\$TMPO	:CLEAR FLAG FOR NEXT REGISTER
2080	014016	022702	170204			CMP	#MAPL01,R2	:SEE IF R2 IS POINTING TO MAPL1
2081	014022	001006				BNE	75\$:BRANCH IF NOT
2082	014024	022712	040000			CMP	#40000,(R2)	:DOES MAPL1 CONTAIN 40000?
2083	014030	001403				BEQ	75\$:BRANCH IF IT DOES
2084	014032	005037	037776			CLR	37776	:CLEAR LOCATION 37776 ONLY - MAPL1 IS TO BE LEFT ALONE
2085	014036	000401				BR	76\$:SKIP OVER REGISTER CLEAR STEP
2086	014040	005012			75\$:	CLR	(R2)	:CLEAR MAP REGISTER JUST TESTED
2087	014042	062700	000200		76\$:	ADD	#200,R0	:POINT TO NEXT MAP REGISTER UNDER TEST
2088	014046	062702	000004			ADD	#4,R2	:POINT TO NEXT MAP REGISTER TO LOAD
2089	014052	022700	177400			CMP	#177400,R0	:SEE IF LAST REGISTER HAS BEEN TRIED
2090	014056	001416				BEQ	9\$:BRANCH TO NEXT SECTION IF SO
2091	014060	023700	001256			CMP	HIGEST,R0	:SEE IF ALL MAP REGS HAVE BEEN TESTED
2092	014064	001222				BNE	1\$:BRANCH IF STILL MORE TO TEST
2093	014066	062700	000200		8\$:	ADD	#200,R0	:POINT TO NEXT MAP REGISTER UNDER TEST
2094	014072	062702	000004			ADD	#4,R2	:POINT TO NEXT MAP REGISTER TO LOAD
2095	014076	022700	177400			CMP	#177400,R0	:SEE IF LAST REGISTER HAS BEEN TRIED
2096	014102	001404				BEQ	9\$:BRANCH TO NEXT SECTION IF SO
2097	014104	020037	001274			CMP	R0,BUPWIN	:SEE IF WE ARE POINTING TO UPPER WINDOW START
2098	014110	001366				BNE	8\$:BRANCH FOR ANOTHER INCREMENT SET IF NOT
2099	014112	000607				BR	1\$:BRANCH BACK FOR ANOTHER RUN
2100	014114	005737	001302		9\$:	TST	ERRCNT	:SEE IF THERE WERE ANY ERRORS
2101	014120	001403				BEQ	TST11	:BRANCH TO NEXT TEST IF NO ERRORS
2102	014122	005337	001110			DEC	\$ERTTL	:DON'T COUNT ERROR +16 AS ANOTHER ERROR
2103	014126	104016				ERROR	+16	:SUMMARY OF DUAL MAPPING ERRORS

2112

.SBTTL TEST 11 - LOAD LOC'S 40000-77776 WITH THEIR ADRES'S

*TEST 11 LOAD LOC'S 40000-77776 WITH THEIR ADRES'S

* THIS TEST IS USED TO LOAD MAIN MEMORY FROM ADDRESS 00040000 TO
 * ADDRESS 000077776 WITH ITS OWN ADDRESS. IT THEN CHECKS THAT
 * MEMORY OVER THE UNIBUS AND LOGS ANG REPORTS ANY ERRORS THAT
 * IT FINDS.

TST11:

014130	014130	000004			SCOPE		
014132	014132	004737	017662		JSR	PC,PRETST	;GO SET UP PRETEST DATA
014136	014136	014362	014252	000011	.WORD	TST12,20\$,11	;DATA USED BY PRETST
2113	014144	042737	000040	172516	BIC	#BIT5,MMR3	;TURN OFF MAP RELOCATION
2114	014152	012737	000400	172350	MOV	#400,KIPAR4	;MAP PAGE 4 TO 8K
2115	014160	012700	040000		MOV	#40000,R0	;STARTING ADDRESS FOR DATA PATTERN
2116	014164	012701	100000		1\$:	MOV #100000,R1	;VIRTUAL ADDRESS
2117	014170	012702	010000		MOV	#4096.,R2	;LOAD 4096 LOCATIONS AT A TIME
2118	014174	010021			2\$:	MOV R0,(R1)+	;LOAD PHY. ADDR. INTO EACH MEMORY LOC.
2119	014176	062700	000002		ADD	#2,R0	;POINT TO NEXT PHYSICAL ADDRESS
2120	014202	077204			SOB	R2,2\$;BRANCH IF 4K OF MEMORY NOT LOADED
2121	014204	062737	000200	172350	ADD	#200,KIPAR4	;POINT TO NEXT 4K BANK OF MEMORY
2122	014212	022737	001000	172350	CMP	#1000,KIPAR4	;SEE IF 16K IS LOADED
2123	014220	101361			BHI	1\$;BRANCH IF MORE MEMORY TO LOAD
2124					;		
2125					;		
2126					;		
2127	014222	022737	171000	172354	CMP	#171000,KIPAR6	;DID I USE ANY MAP REGISTER
2128							;BELOW REGISTER 6 (UB. ADDR 100000)
2129	014230	101454			BLOS	TST12	;BRANCH TO NEXT TEST IF NOT
2130	014232	013700	172354		MOV	KIPAR6,R0	;LOAD PAR6 INTO R0 TO GET
2131							;THE STARTING DATA PATTERN
2132	014236	072027	000006		ASH	#6,R0	;R0 NOW HOLDS THE STARTING DATA PATTERN
2133	014242	012701	140000		3\$:	MOV #140000,R1	;STARTING VIRTUAL ADDRESS
2134	014246	012702	010000		MOV	#4096.,R2	;PREPARE TO READ 4K AT A TIME
2135	014252	011103			20\$:	MOV (R1),R3	;READ MAIN MEMORY THROUGH UNIBUS
2136	014254	020003			CMP	R0,R3	;SEE IF THE ADDRESSES MATCH
2137	014256	001015			BNE	6\$;BRANCH IF ERROR
2138	014260	062701	000002		5\$:	ADD #2,R1	;CHANGE VIRTUAL ADDRESS
2139	014264	062700	000002		ADD	#2,R0	;CHANGE PHYSICAL ADDRESS
2140	014270	077210			SOB	R2,20\$;BRANCH IF 4K OF MEMORY NOT READ
2141	014272	062737	000200	172354	ADD	#200,KIPAR6	;POINT TO NEXT BANK OF 4K THROUGH UNIBUS
2142	014300	022737	171000	172354	CMP	#171000,KIPAR6	;SEE IF THIS POINTS TO 16K PLUS 2
2143	014306	101355			BHI	3\$;BRANCH IF 16K OF MEMORY NOT CHECKED
2144	014310	000416			BR	10\$;TEST FINISHED, BRANCH TO EXIT
2145	014312	013746	172344		6\$:	MOV KIPAR2,-(SP)	;PUT PAR ON STACK FOR ADREXT SUBROUTINE USE
2146	014316	010046			MOV	R0,-(SP)	;PUT VIRTUAL ADDRESS ON STACK FOR ADREXT SUBROUTINE USE
2147	014320	004737	005250		JSR	PC,ADREXT	;GO SET DATA IN THE 4 WORDS OF ADDROR AND ADRAND
2148	014324	010346			MOV	R3,-(SP)	;PUT DATA ON STACK FOR DATEXT SUBROUTINE
2149	014326	004737	005200		JSR	PC,DATEXT	;SUBROUTINE TO LOAD DATAOR AND DATAND

2150 014332 010337 006150
2151 014336 005037 006152
2152 014342 104205
2153 014344 000745
2154 014346 005737 001302
2155 014352 001403

10\$:

MOV R3,EADRS2
CLR EADRS2+2
ERROR +205
BR 5\$
TST ERRCNT
BEQ TST12

M 8

;MOVE DATA IN R3 TO EADRS2 FOR ERROR CALL
;CLEAR UPPER 6 BITS TO PRINT
;DIDN'T READ ADDRESSES CORRECTLY FROM UNIBUS
;CONTINUE TESTING
;WERE THERE ANY ERRORS ON THIS TEST?
;;BRANCH IF NO ERRORS ON THIS TEST

SEQ 0103

2156 014354 005337 001110
2157 014360 104014

DEC \$ERTTL
ERROR +14

:DON'T COUNT ERROR +14 AS ANOTHER ERROR
;SUMMARY OF UNIBUS ADDRESS FAILURES

2166

.SBTTL TEST 12 - MAIN MEMORY TIMEOUT THROUGH MAP
 :*****
 :*TEST 12 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
 :* USABLE MAP REGISTER, WHICH IN THE DEFAULT CASE IS MAP REGISTER
 :* ZERO.
 :*

:*****
 TST12:

014362	014362	000004			SCOPE		
	014364	004737	017662		JSR	PC,PRETST	:GO SET UP PRETEST DATA
	014370	014544	014410	000012	.WORD	TST13,20\$,12	:DATA USED BY PRETST
2167	014376	052737	000040	172516	BIS	#BIT5,MMR3	:TURN MAP RELOCATION BACK ON
2168	014404	005037	001202		CLR	\$TMP4	:CLEAR \$TMP4 FOR ERROR LOOP INDICATOR
2169	014410	004737	014420		20\$: JSR	PC,MMTOTM	:GO DO TEST BELOW
2170	014414	104015			ERROR	+15	:RETURN HERE IF ERROR - UNIBUS DID NOT TIME OUT
2171	014416	000452			BR	TST13	:SKIP OVER THIS PORTION OF THE TEST
2172							
2173	014420	005737	001202		MMTOTM: TST	\$TMP4	:SEE IF THIS ENTRY IS FROM ERROR LOOPING
2174	014424	001402			BEQ	1\$:BRANCH IF NOT
2175	014426	062706	000002		ADD	#2,SP	:CLEAN EXTRA RETURN OFF STACK
2176	014432	005037	001302		1\$: CLR	ERRCNT	:CLEAR ERRCNT FOR THIS TEST
2177	014436	012737	000020	001312	MOV	#TIMOUT,CPUEXP	:EXPECTING TIMEOUT IN THIS TEST
2178	014444	013737	001254	172350	MOV	LOWEST,KIPAR4	:LOAD PAR 4 WITH LOWEST USABLE MAP REG
2179	014452	012777	000074	164620	MOV	#74,@LREGU	:LOAD UPPER 6 BITS OF LOWEST MAP REG
2180	014460	005077	164612		CLR	@LREGL	:LOAD LOWER 16 BITS OF LOWEST MAP REG
2181	014464	005037	001314		20\$: CLR	PCPUER	:CPU ERROR REGISTER LOCATION
2182	014470	013703	100000		MOV	100000,R3	:TRY TO READ THROUGH PAGE 4 THIS REFERENCE WILL GO OUT
2183							:ON THE UNIBUS TO SELECT THE LOWEST USABLE MAP REGISTER (DEFAULT MAP REG. 0). PHYSICAL
2184							:ADDRESS 17700000 IS THEN GENERATED, WHICH SHOULD TIME OUT SINCE IT IS THE FIRST
2185							:NON-EXISTENT LOCATION.
2186	014474	022737	000020	001314	CMP	#TIMOUT,PCPUER	:THE UNIBUS SHOULD HAVE TIMED OUT
2187	014502	001413			BEQ	3\$:BRANCH IF CONDITION WAS CORRECT
2188	014504	105737	007774		TSTB	CPUTYP	:IS THIS AN 11/44?
2189	014510	001403			BEQ	2\$:BRANCH IF IT IS
2190	014512	005737	001302		TST	ERRCNT	:SEE IF A TIMEOUT WAS REPORTED
2191	014516	001005			BNE	3\$:BRANCH IF SO
2192	014520	011646			2\$: MOV	(SP),-(SP)	:PUSH ANOTHER RETURN ONTO THE STACK FOR POSSIBLE ERROR LOOP
2193	014522	012737	000001	001202	MOV	#1,\$TMP4	:SET ERROR LOOP FLAG
2194	014530	000207			RTS	PC	:EXIT
2195	014532	062716	000002		3\$: ADD	#2,(SP)	:CORRECT RETURN PC OVER ERROR CALL
2196	014536	005037	001312		CLR	CPUEXP	:NO CPU TRAPS EXPECTED FOR AWHILE
2197	014542	000207			RTS	PC	:EXIT

2208

```
.SBTTL TEST 13 - RELOC USING LOWEST USABLE MAP REG THRU UNIBUS
:*****
:*TEST 13 RELOC USING LOWEST USABLE MAP REG THRU UNIBUS
:*
:* 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 USABLE 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.
:*
:*****
```

```
014544
014544 000004
014546 004737 017662
014552 014574 014560 000013
2209 014560 005037 001202
2210 014564 004737 005404
2211 014570 104211
2212 014572 000774
```

```
TST13:
SCOPE
JSR PC,PRETST ;GO SET UP PRETEST DATA
.WORD TST14,20$,13 ;DATA USED BY PRETST
CLR $TMP4 ;CLEAR $TMP4 - FLAG USED FOR ERROR RETURN
JSR PC,MAPADD ;GO DO THE TEST
ERROR +211 ;RETURN IS HERE FOR ERROR
BR 1$ ;GO BACK TO THE SUBROUTINE
```

2221

```
.SBTTL TEST 14 - CARRY PROP OF MAP'S RELOC ADDER THRU UNIBUS
*****
:TEST 14 CARRY PROP OF MAP'S RELOC ADDER THRU UNIBUS
:
: EVERY ADDRESS OF THE FORM XXXX0000 IS GENERATED HERE STARTING
: WITH 00030000 UP TO 17000000. THAT IS, THE FIRST OF EVERY 2K
: WORDS IS ADDRESSED, TO INSURE THAT THE ADDER IN THE MAP IS
: WORKING PROPERLY .
:
:*****
```

```
2222 014574 000004
014576 004737 017662
2223
2224
2225
2226
2227 014602 014624 014610 000014
2228 000040
2229 014610 005037 001202
2230 014614 004737 005632
2231 014620 104211
2232 014622 000774
2233 014624
```

```
TST14: SCOPE
JSR PC,PRETST ;GO SET UP PRETEST DATA
*****
:**IMPORTANT** : IF THE POSITION OF THIS TEST IS CHANGED, CHANGE THE THIRD
: WORD BELOW TO THE NEW TEST NUMBER THIS TEST WILL OCCUPY.
:*****
.WORD 2$,20$,14 ;DATA USED BY PRETST
NEXMEM=BIT5 ;BIT05 IS NON-EXISTENT MEMORY BIT IN THE CPU ERROR REGISTER
20$: CLR $TMP4 ;CLEAR $TMP4 - USED AS AN ERROR RETURN FLAG
1$: JSR PC,TCMRA ;GO DO THE TEST
ERROR +211 ;RETURN IS HERE IF AN ERROR
BR 1$ ;RETURN TO THE TEST
2$:
```


2278

.SBTTL TEST 15 - MAIN MEM. T.O. THROUGH MAP, CODE RUN OVER U.B.

*TEST 15 MAIN MEM. T.O. THROUGH MAP, CODE RUN OVER U.B.

*

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

*

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

*

TST15:

014674
 014674 000004
 014676 004737 017662
 014702 014722 014714 000015
 2279 014710 005037 001202
 2280 014714 004737 014420
 2281 014720 104015

SCOPE		
JSR	PC,PRETST	:GO SET UP PRETEST DATA
.WORD	TST16,20\$,15	:DATA USED BY PRETST
CLR	\$TMP4	:CLEAR LOOP ON ERROR FLAG
20\$: JSR	PC,MMTOTM	:GO DO TEST ON PAGE 60 OF THIS LISTING
ERROR	+15	:RETURN HERE IF ERROR - UNIBUS DID NOT TIME OUT

2295

```
.SBTTL TEST 16 - RELOC USING LOWEST USABLE MAP REG USING MAP REG
:*****
:*TEST 16 RELOC USING LOWEST USABLE MAP REG USING MAP 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 USABLE 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 THROUGH
:* THE UNIBUS MAP.
:*****
```

```
C14722
014722 000004
014724 004737 017662
014730 014760 014744 000016
2296 014736 012737 060000 060000
2297
2298 014744 005037 001202
2299 014750 004737 005404
2300 014754 104211
2301 014756 000774
```

```
TST16:
SCOPE
JSR PC,PRETST ;GO SET UP PRETEST DATA
.WORD TST17,20$,16 ;DATA USED BY PRETST
MOV #060000,060000 ;MAKE SURE THAT ADDRESS 060000
;CONTAINS ITS OWN ADDRESS AS DATA
20$: CLR $TMP4 ;CLEAR $TMP4 - USED IN ERROR RETURN
1$: JSR PC,MAPADD ;GO DO THE TEST
ERROR +211 ;RETURN IS HERE FOR ERROR
BR 1$ ;RETURN TO THE SUBROUTINE
```

2310

```
.SBTTL TEST 17 - CARRY PROP OF MAP'S RELOC ADDER USING MAP REG
:*****
:*TEST 17 CARRY PROP OF MAP'S RELOC ADDER USING MAP REG
:*
:* EVERY ADDRESS OF THE FORM XXXX0000 IS GENERATED HERE STARTING
:* WITH 00030000 UP TO 17000000. THAT IS THE FIRST OF EVERY 2K
:* WORDS IS ADDRESSED, TO INSURE THAT THE ADDER IN THE MAP IS
:* WORKING PROPERLY .
:*****
```

```
014760
014760 000004
014762 004737 017662
014766 015010 014774 000017
2311 014774 005037 001202
2312 015000 004737 005632
2313 015004 104211
2314 015006 000774
```

```
TST17:
SCOPE
JSR PC,PRETST ;GO SET UP PRETEST DATA
.WORD TST20,20$,17 ;DATA USED BY PRETST
CLR $TMP4 ;CLEAR $TMP4 - USED AS ERROR RETURN FLAG
JSR PC,TCPMRA ;GO DO THE TEST
ERROR +211 ;RETURN IS HERE IF AN ERROR
BR 1$ ;RETURN TO THE TEST
```


2345

```
.SBTTL TEST 20 - VERIFY TRAP DUE TO CACHE PARITY INTERRUPT
:*****
:*TEST 20 VERIFY TRAP DUE TO CACHE PARITY INTERRUPT
:*****
:*
:*****NOTE*****
:* THE MAP WILL BE SHUT OFF FOR THE REMAINDER OF THE DIAGNOSTIC
:* BEFORE ANY TEST CODE IS EXECUTED. IT IS NOT NEEDED.
:*****
:*
:* THIS TEST IS OPTIONAL AND IS SELECTED BY SETTING MFM HARDWARE
:* SWITCH REGISTER BIT 08 TO A 1 IN THE CASE OF STANDALONE OPERATION
:* OF THE DIAGNOSTIC. IN THE CASE OF MANUFACTURING APT RUNTIME MODE
:* THEN BIT 08 OF $SWREG IS SET TO 1 THROUGH APT SCRIPTING.
:*
:* THE TEST VERIFIES THE SIGNAL GENERATED FROM THE CACHE
:* TO THE UBI MODULE WHICH INDICATES TO THE UBI THAT A CACHE INTERRUPT
:* IS BEING CALLED FOR(CACHE PE INTR L).
:*
:* THIS TEST ASSUMES THAT ALL MODULES EXCEPT THE UBI MODULE ARE KNOWN
:* GOOD MODULES.
:*
:* THIS TEST TOGETHER WITH OTHER CACHE TESTS,ALLOW MFG. TO ELIMINATE
:* HAVING TO RUN THE CACHE DIAGNOSTIC DURING QUICK
:* VERIFY TESTING OF THE UBI MODULE.
:*
:* TEST DESCRIPTION:
:* VERIFY INTERRUPT LOGIC BY ASSURING THAT A TRAP OCCURS TO LOCATION
:* 114 WHEN A LOCATION PREVIOUSLY WRITTEN
:* WITH WRONG HI/LO BYTE PARITY IS ACCESSED.
:* CONDITIONS: PEA=0
:* DCPI=0
:*****
```

```
015010
015010 000004
015012 004737 017662
015016 015352 015116 000020
2346 015024 005037 177572
2347 015030 005037 172516
2348 015034 005037 172340
2349 015040 012737 000200 172342
2350 177746
2351 177744
2352 177750
2353 000114
2354 000116
2355 015046 105737 007774
2356 015052 001404
2357 015054 105237 001100
2358 015060 000137 015706
2359 015064 132737 000200 020035 21$
2360 015072 001405
```

```
TST20:
SCOPE
JSR PC,PRETST ;GO SET UP PRETEST DATA
.WORD TST21,20$,20 ;DATA USED BY PRETST
CLR MMRO ;TURN OFF MEMORY MANAGEMENT
CLR MMR3 ;TURN OFF MAP RELOCATION
CLR KIPAR0 ;PUT PAR0 BACK WHERE IT SHOULD AND
MOV #200,KIPAR1 ;PUT PAR1 BACK WHERE IT SHOULD
CACHE =177746 ;CACHE LOCATION =177746
CMPE =177744 ;CMPE LOCATION =177744
MAINT =177750 ;MAINTENANCE LOCATION =177750
CTRAMPV =114 ;CACHE TRAP VECTOR =114
CTRAPS =116 ;CACHE TRAP STATUS =116
TSTB CPUTYP ;IS THIS AN 11/24?
BEQ 21$ ;BRANCH OVER JUMP IF 11/44
INCB $TSTNM ;INCREMENT TEST NUMBER TO SIMULATE SKIPPING TEST
JMP TST22 ;JUMP OVER THIS AND NEXT TESTS
BITB #200,$ENVM ;IS APT SIZING?
BEQ 12$ ;NO;TRY HARDWARE SWITCH REGISTER
```

2361	015074	032737	000100	020036		BIT	#100,\$SWREG	:YES APT IS SIZING;DOES APT SAY TO DO THIS TEST	
2362	015102	001523				BEQ	TST21	::NO - SKIP TEST	SEQ 0113
2363	015104	000404				BR	20\$:YES,DO TEST	
2364	015106	032777	000100	164034	12\$:	BIT	#100,@SWR	:DOES HARDWARE SWITCH REGISTER SAY TO DO TEST?	
2365	015114	001516				BEQ	TST21	::NO - SKIP TEST	
2366	015116	042737	000001	177572	20\$:	BIC	#1,MMRO	:TURN OFF RELOCATION	

2446

```

.SBTTL TEST 21 - VERIFY TRAP DUE TO CACHE PARITY ABORT
*****
*TEST 21 VERIFY TRAP DUE TO CACHE PARITY ABORT
* THIS TEST IS OPTIONAL AND IS SELECTED BY SETTING MFM HARDWARE
* SWITCH REGISTER BIT 08 TO A 1 IN THE CASE OF STANDALONE OPERATION
* OF THE DIAGNOSTIC. IN THE CASE OF MANUFACTURING APT RUNTIME MODE
* THEN BIT 08 OF $$SWREG IS SET TO 1 THROUGH APT SCRIPTING.
*
* THE TEST VERIFIES THE SIGNAL GENERATED FROM THE CACHE(BUS PBL)
* TO THE UBI MODULE WHICH INDICATES TO THE UBI THAT A CACHE ABORT
* IS BEING CALLED FOR.
*
* THIS TEST ASSUMES THAT ALL MODULES EXCEPT UBI ARE KNOWN GOOD MODULES
*
* THIS TEST TOGETHER WITH OTHER CACHE TESTS,ALLOW MFG. TO ELIMINATE
* HAVING TO RUN THE CACHE DIAGNOSTIC DURING QUICK
* VERIFY TESTING OF THE UBI MODULE.
*
* TEST DESCRIPTION:
*
* VERIFY ABORT LOGIC BY THE FOLLOWING RESULTS WHEN A LOCATION
* PREVIOUSLY WRITTEN WITH WRONG HI/LO BYTE PARITY IS ACCESSED.
* 1. INSTRUCTION CYCLE WILL BE ABORTED
* 2. THE ABORT CAUSES TRAP TO 114
*
* PROCEDURE: INHIBIT CLOCKING OF PARITY ERROR SIGNAL TO
* INTERRUPT LOGIC. ALLOW CMPE<15> TO BE SET
* BY ABORT SIGNAL WHICH IS ASSERTED BY PARITY
* ERROR SIGNAL TO ABORT LOGIC.
*
* CONDITIONS: PEA=1
* DCPI=1
    
```

```

015352
015352 000004
015354 004737 017662
015360 015706 015420 000021
2447 015366 132737 000200 020035
2448 015374 001405
2449 015376 032737 000100 020036
2450 015404 001533
2451 015406 000404
2452 015410 032737 000100 177570 12$
2453 015416 001526
2454 015420 042737 000001 177572 20$
2455 015426 052737 000400 177746
2456 015434 032737 010000 177746 2$
2457 015442 001374
2458 015444 013702 000000
    
```

```

*****
TST21:
SCOPE
JSR PC,PRETST ;GO SET UP PRETEST DATA
.WORD TST22,20$,21 ;DATA USED BY PRETST
BITB #200,$ENVM ;IS APT SIZING?
BEQ 12$ ;NO;TRY HARDWARE SWITCH REGISTER
BIT #100,$SWREG ;YES APT IS SIZING;DOES APT SAY TO DO THIS TEST
BEQ 21$ ;NO - SKIP TEST
BR 20$ ;YES,DO TEST
BIT #100,177570 ;DOES HARDWARE SWITCH REGISTER SAY TO DO TEST?
BEQ 21$ ;NO - SKIP TEST
BIC #1,177572 ;TURN OFF RELOCATION
BIS #400,CACHE ;FLUSH CACHE TO INVALIDATE ALL CACHE LOCATIONS
BIT #10000,CACHE ;WAIT TILL DONE
BNE 2$
MOV 0,R2 ;SAVE ADDRESS 0 CONTENTS
    
```

2459 015450 005037 000000
2460 015454 005003
2461 015456 013700 000114
2462 015462 013701 000116
2463 015466 012737 015576 000114
2464 015474 012737 000340 000116

CLR 0
CLR R3
MOV CTRAPV,R0
MOV CTRAPS,R1
MOV #48,CTRAPV
MOV #340,CTRAPS

M 9

;ALL 0'S TO LOCATION 0
;ADDRESS 0 TO R3
;SAVE VECTORS
;SETUP FOR TRAP

SEQ 0116

```

2465 015502 112737 000002 177750      MOVB    #2,MAINT      ;HODO ALLOWS CACHE UPDATES AND CLOCKING OF
2466                                     ;PARITY INFO TO INTERRUPT LOGIC ONLY DURING
2467                                     ;THE DESTINATION ACCESS OF AN INSTRUCTION.
2468 015510 005005      CLR     R5            ;CLEAR ERROR FLAG
2469 015512 012704 177777      MOV     #-1,R4        ;ALL 1'S TO R4
2470 015516 012737 000015 177746      MOV     #15,CACHE     ;NO UCB SO AS TO WRITE CACHE STORES
2471 015524 005737 040000      TST    40000         ;UPDATE CACHE LOCATION 0000 WITH CORRECT PARITY
2472                                     ;STORAGE
2473 015530 052737 000100 177746      BIS    #100,CACHE     ;ALLOW WRITE WRONG PARITY DATA TO LO & HI BYTE
2474                                     ;PARITY STORE.
2475 015536 005713      TST    (R3)          ;READ UPDATE TO CACHE LOCATION 0000 WRITE WRONG
2476                                     ;PARITY TO HI/LO BYTE PARITY STORES
2477 015540 042737 000100 177746      BIC    #100,CACHE     ;DISABLE WWP
2478 015546 005037 177744      CLR    CMPE          ;CLEAR CMPE AND PARITY DETECT LOGIC
2479 015552 042737 000004 177746      BIC    #4,CACHE       ;ENABLE LOW CACHE
2480 015560 052737 000200 177746      BIS    #200,CACHE     ;ALLOW FOR ABORT
2481 015566 011304      MOV    (R3),R4       ;READ HIT LO & HI BYTE PARITY CHECK GENERATORS WILL
2482 015570 000240      NOP                 ;NEEDED IN 11/44 TO ALLOW 1 INSTRUCTION BEFORE ABORT
2483                                     ;DETECT WRONG PARITY USING HODO AND SOURCE MODE FOR READING LOCATION 0 WILL
2484                                     ;INHIBIT PARITY ERROR FROM BEING CLOCKED TO INTERRUPT LOGIC. HOWEVER, THE PARITY
2485                                     ;ERROR SIGNAL WILL CAUSE THE ABORT SIGNAL TO BE ASSERTED. THE ABORT SIGNAL WILL
2486                                     ;BECAUSE CMPE<15> TO BE SET. THIS INSTRUCTION SHOULD BE ABORTED
2487 015572 005205      INC    R5            ;INDICATE NO TRAP OCCURED
2488 015574 000401      BR    5$            ;BRANCH OVER STACK CORRECTION
2489 015576 022626      4$:  CMP    (R6)+,(R6)+ ;READJUST STACK
2490 015600 005037 177744      5$:  CLR    CMPE          ;CLEAR CMPE
2491 015604 012737 001015 177746      MOV    #1015,CACHE    ;DISABLE CACHE
2492 015612 105037 177750      CLRB  MAINT          ;DISABLE MAINT. MODE
2493 015616 010237 000000      MOV    R2,0          ;RESTORE VECTORS
2494 015622 010037 000114      MOV    R0,CTRAPV
2495 015626 010137 000116      MOV    R1,CTRAPS
2496 015632 052737 000400 177746      BIS    #400,CACHE     ;BEFORE LEAVING TEST FLUSH CACHE TO
2497                                     ;ELIMINATE ANY EFFECTS OF WWP
2498 015640 032737 010000 177746 30$:  BIT    #10000,CACHE   ;WAIT TILL DONE
2499 015646 001374      BNE   30$
2500 015650 022704 177777      8$:  CMP    #-1,R4        ;WAS INSTRUCTION ABORTED LEAVING R4 INTACT?
2501 015654 001401      BEQ   9$            ;YES
2502 015656 104021      ERROR +21          ;INTERRUPT/ABORT TESTS R4 WAS OVERWRITTEN WITH
2503                                     ;DATA INDICATING THAT INSTRUCTION WAS NOT ABORTED
2504 015660 005705      9$:  TST    R5            ;DID TRAP OCCUR
2505 015662 001401      BEQ   10$          ;YES, PASS
2506 015664 104022      ERROR +22          ;INTERRUPT/ABORT TESTS TRAP DID NOT OCCUR DUE TO ABORT
2507 015666 012737 000000 177746 10$:  MOV    #0,CACHE       ;TURN CACHE ON
2508 015674 062737 000007 001100 21$:  ADD    #7,$STSTM      ;ADD 7 TO $STSTM TO COMPENSATE FOR 7 TESTS SKIPPED
2509 015702 000137 017202      JMP    TST31         ;JUMP OVER NEXT 7 TESTS - THEY ARE FOR AN 11/24 ONLY

```

2522

```
.SBTTL TEST 22 - LMA REGISTER PHYSICAL ADDRESS CHECK
:*****
:*TEST 22 LMA REGISTER PHYSICAL ADDRESS CHECK
:*
:* THE NEXT 7 TESTS ARE EXECUTED ON THE 11/24 ONLY.
:*
:* THIS TEST IS TO CHECK OUT THE LMA (LAST MAPPED ADDRESS) REGISTER FOR
:* PROPER CONTENTS. FIRST, THE PAR AND MAP REGISTERS ARE SET, THEN A
:* PHYSICAL ADDRESS IS LOADED INTO AN EXPECTED DATA LOCATION. THEN THE
:* MAP IS INSURED TO BE ON AND A MEMORY ACCESS IS DONE, USING THE MAP
:* REGISTER SO THE LMA IS LOADED. THE LMA IS THEN CHECKED FOR CONTAINING
:* THE PROPER CONTENTS, CALLING AN ERROR IF EXPECTED DATA DID NOT APPEAR.
:*****
```

```
TST22:
SCOPE
JSR PC,PRETST ;GO SET UP PRETEST DATA
.WORD TST23,20$,22 ;DATA USED BY PRETST
20$: BIC #BIT0,MMR0 ;TURN OFF MEMORY MANAGEMENT
BIC #60,MMR3 ;TURN OFF 22-BIT AND MAP RELOCATION
MOV #20$,EADRES ;LOAD EADRES WITH LOWER 16 BITS OF THE PHYSICAL ADDRESS
CLR EADRES+2 ;LOAD UPPER 6 BITS WITH PHYSICAL BITS EXPECTED
MOV #20$+BIT13,R0 ;MOVE ADDRESS +20000 (TO REFERENCE PAR1) TO R0
CLR MAPLO ;CLEAR MAP 0 LOWER 16
CLR MAPHO ;CLEAR MAP 0 UPPER 6
MOV KIPAR0,-(SP) ;SAVE KIPAR0
MOV KIPAR1,-(SP) ;SAVE KIPAR1
CLR KIPAR0 ;CLEAR PAR0 FOR NO MAP USE FOR THIS AREA
MOV #170000,KIPAR1 ;PUT 170000 IN PAR1 FOR PROPER MEMORY REFERENCE
BIS #60,MMR3 ;TURN ON 22-BIT AND MAP RELOCATION
BIS #BIT0,MMR0 ;TURN ON MEMORY MANAGEMENT
MOV (R0),R1 ;DO THE MAP REGISTER READ THROUGH THE MAP
CMP EADRES,LMALOW ;SEE IF LMA LOWER 16 WERE LOADED PROPERLY
BNE 1$ ;BRANCH TO CALL ERROR IF NOT
MOV LMAHI,$TMPO ;MOVE HI 6 BITS TO $TMPO FOR PREPARATION
BIC #177700,$TMPO ;CLEAR ALL BUT LOWER 6 BITS
BEQ 2$ ;BRANCH AROUND ERROR IF OK
2542 016052 013737 177734 006150 1$: MOV LMALOW,EADRS2 ;MOVE LOWER 16 BITS OF RECEIVED DATA TO EADRS2 FOR ERROR
2543 016060 013737 177736 006152 MOV LMAHI,EADRS2+2 ;MOVE UPPER 6 BITS OF RECEIVED DATA TO EADRS2+2 FOR ERROR
2544 016066 012637 172342 MOV (SP)+,KIPAR1 ;RESTORE KIPAR1
2545 016072 012637 172340 MOV (SP)+,KIPAR0 ;RESTORE KIPAR0
2546 016076 104023 ERROR +23 ;LMA NOT LOADED PROPERLY
2547 016100 000404 BR TST23 ;BRANCH OVER PAR RESTORATION - NOT NEEDED
2548 016102 012637 172342 2$: MOV (SP)+,KIPAR1 ;RESTORE KIPAR1
2549 016106 012637 172340 MOV (SP)+,KIPAR0 ;RESTORE KIPAR0
```

2556

```
.SBTTL TEST 23 - LMA FORCE JUMPER BIT TEST  
:*****  
:*TEST 23 LMA FORCE JUMPER BIT TEST  
:*  
:* THIS TEST DETERMINES THAT THE FORCE JUMPER BIT OF THE LMA IS ZERO AFTER  
:* A SYSTEM RESET.  
:*  
:*****
```

```
016112  
016112 000004  
016114 004737 017662  
016120 016152 016126 000023  
2557 000100  
2558 016126 000005  
2559 016130 032737 000100 177736  
2560 016136 001405  
2561 016140 013701 177736  
2562 016144 042701 000100  
2563 016150 104024
```

```
TST23:  
SCOPE  
JSR PC,PRETST ;GO SET UP PRETEST DATA  
.WORD TST24,20$,23 ;DATA USED BY PRETST  
FJBIT =100 ;FORCE JUMPER BIT IS BIT 6  
20$: RESET ;RESET THE WORLD, CLEARING THE FJBIT  
BIT #FJBIT,LMAHI ;CHECK THE BIT FOR BEING ZERO  
BEQ TST24 ;:BRANCH TO NEXT TEST IF OK  
MOV LMAHI,R1 ;MOVE LMAHI TO R1 FOR ERROR CALL  
BIC #FJBIT,R1 ;CLEAR THE BIT THAT SHOULD HAVE BEEN CLEAR  
ERROR +24 ;LMA FORCE JUMPER BIT NOT ZERO
```


2573

.SBTTL TEST 24 - SETTING LMA FORCE JUMPER BIT TEST

*TEST 24 SETTING LMA FORCE JUMPER BIT TEST

*

* THIS TEST SETS THE FORCE JUMPER BIT AND TESTS ITS FUNCTIONALITY IF
 * THE JUMPERS ARE NOT IN THEIR DEFAULT STATE. IF NOT ('LOWEST' OR
 * 'HIGEST' DO NOT CONTAIN THE DEFAULT VALUES OF 170000 OR 177400
 * RESPECTIVELY), THIS TEST INSURES THAT THE PREVIOUSLY DISABLED MAP
 * REGISTERS ARE ENABLED WITH THE FJ BIT SET.

*

TST24:

016152	016152	000004				SCOPE		
016154	004737	017662				JSR	PC,PRETST	;GO SET UP PRETEST DATA
016160	016352	016166	000024			.WORD	TST25,20\$,24	;DATA USED BY PRETST
2574	016166	052737	000100	177736	20\$:	BIS	#FJBIT,LMAHI	;SET THE BIT
2575	016174	032737	000100	177736		BIT	#FJBIT,LMAHI	;SEE IF IT WAS SET
2576	016202	001005				BNE	1\$;BRANCH IF SET
2577	016204	013701	177736			MOV	LMAHI,R1	;MOVE LMAHI TO R1 FOR ERROR CALL
2578	016210	052701	000100			BIS	#FJBIT,R1	;SET THE BIT THAT SHOULD HAVE BEEN SET
2579	016214	104025				ERROR	+25	;LMA FORCE JUMPER BIT NOT SET
2580	016216	022737	170000	001254	1\$:	CMP	#170000,LOWEST	;SEE IF MAP REGISTER 0 IS LOWEST
2581	016224	001004				BNE	2\$;BRANCH AROUND UPPER LIMIT CHECK IF NOT
2582	016226	022737	177400	001256		CMP	#177400,HIGEST	;SEE IF MAP REGISTER 31 IS HIGEST
2583	016234	001446				BEQ	TST25	;BRANCH TO NEXT TEST IF SO
2584	016236	012737	016276	001106	2\$:	MOV	#3\$,\$LPERR	;RESET LOOP ON ERROR TO 3\$
2585	016244	013746	172350			MOV	KIPAR4,-(SP)	;SAVE PAR4
2586	016250	013737	001260	172350		MOV	UBMLOW,KIPAR4	;MOVE LOWEST PAGE OF MEMORY WINDOW TO PAR4
2587	016256	162737	000200	172350		SUB	#200,KIPAR4	;SUBTRACT 200 FROM KIPAR4 TO PREPARE FOR SUBROUTINE
2588	016264	012700	117776			MOV	#117776,R0	;THIS WILL BE USED TO SELECT PAR4, ADDRESS 17776
2589	016270	012702	125252			MOV	#125252,R2	;LOAD CONSTANT TO R2
2590	016274	000417				BR	5\$;BRANCH OVER LOOP SETUP
2591	016276	032777	001000	162644	3\$:	BIT	#BIT9,@SWR	;SEE IF LOOP ON ERROR IS STILL SET
2592	016304	001407				BEQ	4\$;BRANCH OUT IF NOT
2593	016306	162737	000200	172350		SUB	#200,KIPAR4	;UNDO NEXT PAGE MAP FOR LOOPING
2594	016314	004737	005046			JSR	PC,TSTLOC	;GO TEST LOCATION FOR WRITEABILITY
2595	016320	000766				BR	3\$;BRANCH BACK FOR LOOPBACK
2596	016322	104027				ERROR	+27	;FORCE JUMPER BIT FAILS TO REVERT MAP REGISTER STATUS TO DEF
2597	016324	023737	001262	172350	4\$:	CMP	UBMHI,KIPAR4	;SEE IF HIGEST HAS BEEN REACHED
2598	016332	001405				BEQ	6\$;BRANCH TO RESTORE PAR4 AND LEAVE TEST IF SO
2599	016334	004737	005046		5\$:	JSR	PC,TSTLOC	;GO TEST LOCATION FOR WRITEABILITY
2600	016340	000771				BR	4\$;BRANCH BACK FOR ANOTHER TEST IF LOCATION LOADED
2601	016342	104027				ERROR	+27	;FORCE JUMPER BIT FAILS TO REVERT MAP REGISTER STATUS TO DEF
2602	016344	000767				BR	4\$;BRANCH BACK FOR ANOTHER TEST
2603	016346	012637	172350		6\$:	MOV	(SP)+,KIPAR4	;RESTORE KIPAR4

2609

```
.SBTTL TEST 25 - CLEARING THE FORCE JUMPER BIT
:*****
:*TEST 25 CLEARING THE FORCE JUMPER BIT
:*
:* THIS TEST CLEARS THE FJ BIT AND INSURES THAT IT IS SUCCESSFULLY CLEARED.
:*
:*****
```

```
016352
016352 000004
016354 004737 017662
2610 016360 016416 016366 000025
2611 016366 042737 000100 177736
2612 016374 032737 000100 177736
2613 016402 001405
2614 016404 013701 177736
2615 016410 042701 000100
016414 104024
```

```
TST25:
SCOPE
JSR PC,PRETST ;GO SET UP PRETEST DATA
.WORD TST26,20$,25 ;DATA USED BY PRETST
BIC #FJBIT,LMAHI ;CLEAR THE BIT
BIT #FJBIT,LMAHI ;CHECK TO SEE THAT IT WAS CLEARED
BEQ TST26 ;;BRANCH IF CLEARED
MOV LMAHI,R1 ;MOVE LMAHI TO R1 FOR ERROR CALL
BIC #FJBIT,R1 ;CLEAR THE BIT THAT SHOULD HAVE BEEN CLEAR
ERROR +24 ;LMA FORCE JUMPER BIT NOT ZERO
```

2622

```
.SBTTL TEST 26 - LMA CONTROL BITS TEST - DATI
:*****
:*TEST 26          LMA CONTROL BITS TEST - DATI
:*
:*          THIS TEST INSURES THE CONTROL BITS 14 AND 15 LOAD PROPERLY DOING A
:*          DATI.
:*
:*****
```

```
TST26:
SCOPE
JSR    PC,PRETST          ;GO SET UP PRETEST DATA
.WORD  TST27,20$,26      ;DATA USED BY PRETST
MOV    KIPAR0,-(SP)       ;SAVE KIPAR0
MOV    KIPAR1,-(SP)       ;SAVE KIPAR1
MOV    #170000,KIPAR1     ;LOAD KIPAR1
CLR    KIPAR0             ;CLEAR KIPAR0
MOV    #1,MMR0            ;TURN ON MEMORY MANAGEMENT
MOV    #20,MMR3           ;TURN ON 22-BIT ADDRESSING
MOV    #1$+BIT13,R0       ;MOVE LOCATION ADDRESS +20000 (TO REF PAR1) TO R0
MOV    (R0),R1            ;DO A DATI
MOV    LMAHI,R2           ;MOVE LMAHI TO R2 FOR CONTROL
BIC    #37777,R2          ;CLEAR ALL BUT THE CONTROL BITS
BEQ    3$                 ;BRANCH TO FINISH TEST IF ALL CLEAR
CLR    $TMP0              ;CLEAR THE EXPECTED LOCATION
BR     2$                 ;GO CALL ERROR

20$:
MOV    KIPAR0,-(SP)       ;SAVE KIPAR0
MOV    KIPAR1,-(SP)       ;SAVE KIPAR1
MOV    #170000,KIPAR1     ;LOAD KIPAR1
CLR    KIPAR0             ;CLEAR KIPAR0
MOV    (R0),R1            ;DO A DATI
MOV    LMAHI,R2           ;MOVE LMAHI TO R2 FOR CONTROL
BIC    #37777,R2          ;CLEAR ALL BUT THE CONTROL BITS
CMP    R2,$TMP0           ;SEE IF EXPECTED CAME UP
BNE    2$                 ;BRANCH TO CALL ERROR IF IT DIDN'T
BIT    #BIT9,@SWR         ;SEE IF LOOP ON ERROR IS SET
BNE    1$                 ;BRANCH BACK FOR ANOTHER TRY IF SET
BR     3$                 ;GO EXIT TEST - ALL DONE

1$:
MOV    (SP)+,KIPAR1       ;RESTORE KIPAR1
MOV    (SP)+,KIPAR0       ;RESTORE KIPAR0
ERROR  +26                ;LMA CONTROL BITS INCORRECT
BR     TST27              ;BRANCH OVER PAR RESTORATIONS

2$:
MOV    (SP)+,KIPAR1       ;RESTORE KIPAR1
MOV    (SP)+,KIPAR0       ;RESTORE KIPAR0

3$:
MOV    (SP)+,KIPAR1       ;RESTORE KIPAR1
MOV    (SP)+,KIPAR0       ;RESTORE KIPAR0
```

2654

```
.SBTTL TEST 27 - LMA CONTROL BITS TEST - DATO
*****
*TEST 27          LMA CONTROL BITS TEST - DATO
*
*   THIS TEST INSURES THE CONTROL BITS 14 AND 15 LOAD PROPERLY DOING A
*   DATO.
*
*****
```

```
TST27:
SCOPE
JSR    PC,PRETST      ;GO SET UP PRETEST DATA
.WORD  TST30,20$,27  ;DATA USED BY PRETST
=100000
DATO
2655   016616 000004
2656   016616 004737 017662
2656   016624 017010 016710 000027
2656   016632 013746 172340
2656   016636 013746 172342
2656   016642 012737 170000 172342
2656   016650 005037 172340
2657   016654 012700 036732
2658   016660 010110
2659   016662 013702 177736
2660   016666 042702 037777
2661   016672 022702 100000
2662   016676 001440
2663   016700 012737 100000 001172
2664   016706 000426
2665   016710
2665   016710 013746 172340
2665   016714 013746 172342
2665   016720 012737 170000 172342
2665   016726 005037 172340
2666   016732 010110
2667   016734 013702 177736
2668   016740 042702 037777
2669   016744 020237 001172
2670   016750 001005
2671   016752 032777 001000 162170
2672   016760 001364
2673   016762 000406
2674   016764 012637 172342
2675   016770 012637 172340
2676   016774 104026
2677   016776 000404
2678   017000 012637 172342
2679   017004 012637 172340

MOV    KIPARO,-(SP)   ;SAVE KIPARO
MOV    KIPAR1,-(SP)  ;SAVE KIPAR1
MOV    #170000,KIPAR1 ;LOAD KIPAR1
CLR    KIPARO        ;CLEAR KIPARO
MOV    #1$+BIT13,R0  ;MOVE LOCATION ADDRESS +20000 (TO REF PAR1) TO R0
MOV    R1,(R0)       ;DO A DATO
MOV    LMAHI,R2      ;MOVE LMAHI TO R2 FOR CONTROL BIT ANALYSIS
BIC    #37777,R2     ;CLEAR ALL BUT THE CONTROL BITS
CMP    #DATO,R2      ;SEE IF BIT 15 IS SET AND 14 IS CLEAR
BEQ    3$            ;BRANCH IF OK
MOV    #DATO,$TMP0   ;MOVE EXPECTED DATA TO $TMP0
BR     2$            ;GO CALL ERROR

20$:
MOV    KIPARO,-(SP)   ;SAVE KIPARO
MOV    KIPAR1,-(SP)  ;SAVE KIPAR1
MOV    #170000,KIPAR1 ;LOAD KIPAR1
CLR    KIPARO        ;CLEAR KIPARO
MOV    R1,(R0)       ;DO A DATO
MOV    LMAHI,R2      ;MOVE LMAHI TO R2 FOR CONTROL
BIC    #37777,R2     ;CLEAR ALL BUT THE CONTROL BITS
CMP    R2,$TMP0      ;SEE IF EXPECTED CAME UP
BNE    2$            ;BRANCH TO CALL ERROR IF IT DIDN'T
BIT    #BIT9,@SWR    ;SEE IF LOOP ON ERROR IS SET
BNE    1$            ;BRANCH BACK FOR ANOTHER TRY IF SET
BR     3$            ;GO EXIT - ALL DONE

1$:
MOV    (SP)+,KIPAR1  ;RESTORE KIPAR1
MOV    (SP)+,KIPAR0  ;RESTORE KIPAR0
ERROR  +26           ;LMA CONTROL BITS INCORRECT
BR     TST30         ;;BRANCH OVER PAR RESTORATION - NOT NEEDED

2$:
MOV    (SP)+,KIPAR1  ;RESTORE KIPAR1
MOV    (SP)+,KIPAR0  ;RESTORE KIPAR0
```

2686

```
.SBTTL TEST 30 - LMA CONTROL BITS TEST - DATOB
:*****
:*TEST 30          LMA CONTROL BITS TEST - DATOB
:*
:*      THIS TEST INSURES THE CONTROL BITS 14 AND 15 LOAD PROPERLY DOING A
:*      DATOB.
:*
:*****
```

```
017010
017010 000004
017012 004737 017662
017016 017202 017102 000030
2687 140000
2688 017024 013746 172340
017030 013746 172342
017034 012737 170000 172342
017042 005037 172340
2689 017046 012700 037124
2690 017052 110110
2691 017054 013702 177736
2692 017060 042702 037777
2693 017064 022702 140000
2694 017070 001440
2695 017072 012737 140000 001172
2696 017100 000426
2697 017102
017102 013746 172340
017106 013746 172342
017112 012737 170000 172342
017120 005037 172340
2698 017124 110110
2699 017126 013702 177736
2700 017132 042702 037777
2701 017136 020237 001172
2702 017142 001005
2703 017144 032777 001000 161776
2704 017152 001364
2705 017154 000406
2706 017156 012637 172342
2707 017162 012637 172340
2708 017166 104026
2709 017170 000404
2710 017172 012637 172342
2711 017176 012637 172340
```

```
TST30:
SCOPE
JSR PC,PRETST ;GO SET UP PRETEST DATA
.WORD TST31,20$,30 ;DATA USED BY PRETST
DATOB =140000 ;DATOB CONTROL BITS STATUS=140000
MOV KIPARO,-(SP) ;SAVE KIPARO
MOV KIPAR1,-(SP) ;SAVE KIPAR1
MOV #170000,KIPAR1 ;LOAD KIPAR1
CLR KIPARO ;CLEAR KIPARO
MOV #1$+BIT13,R0 ;MOVE LOCATION ADDRESS +20000 (TO REF PAR1) TO R0
MOVB R1,(R0) ;DO A DATOB
MOV LMAHI,R2 ;MOVE LMAHI TO R2 FOR CONTROL
BIC #37777,R2 ;CLEAR ALL BUT THE CONTROL BITS
CMP #DATOB,R2 ;SEE IF BIT 15 AND 14 ARE SET
BEQ 3$ ;BRANCH IF OK
MOV #DATOB,$TMP0 ;MOVE EXPECTED DATA TO $TMP0
BR 2$ ;BRANCH TO CALL ERROR

20$:
MOV KIPARO,-(SP) ;SAVE KIPARO
MOV KIPAR1,-(SP) ;SAVE KIPAR1
MOV #170000,KIPAR1 ;LOAD KIPAR1
CLR KIPARO ;CLEAR KIPARO
1$:
MOVB R1,(R0) ;DO A DATOB
MOV LMAHI,R2 ;MOVE LMAHI TO R2 FOR CONTROL
BIC #37777,R2 ;CLEAR ALL BUT THE CONTROL BITS
CMP R2,$TMP0 ;SEE IF EXPECTED CAME UP
BNE 2$ ;BRANCH IF IT DIDN'T
BIT #BIT9,@SWR ;SEE IF LOOP ON ERROR IS SET
BNE 1$ ;BRANCH BACK FOR ANOTHER TRY IF SET
BR 3$ ;BRANCH TO COMPLETE TEST - ALL DONE

2$:
MOV (SP)+,KIPAR1 ;RESTORE KIPAR1
MOV (SP)+,KIPARO ;RESTORE KIPARO
ERROR +26 ;LMA CONTROL BITS INCORRECT
BR TST31 ;BRANCH OVER PAR RESTORATION - NOT NEEDED

3$:
MOV (SP)+,KIPAR1 ;RESTORE KIPAR1
MOV (SP)+,KIPARO ;RESTORE KIPARO
```

2712
2713
2714
2715
2716
2717
2718

```
:*****
:*
:*      >>NOTE<<: 'DATIP' CANNOT BE CHECKED IN THE 11/24 BECAUSE THE LMA IS
:*      WRITTEN TWICE WHEN A 'DATIP' IS EXECUTED, DESTROYING THE
:*      'DATIP' STATE THAT WAS WRITTEN FIRST.
:*
:*****
```

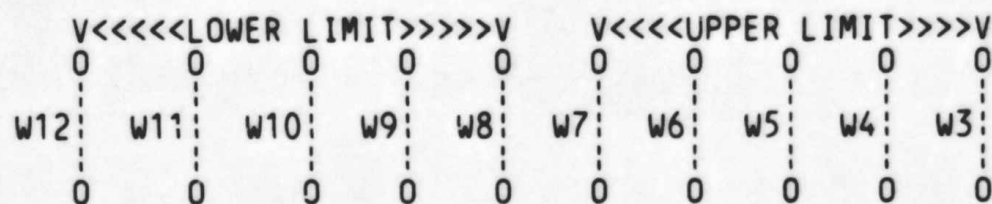
2751

.SBTTL TEST 31 - MEMORY ON UNIBUS TEST

 *TEST 31 MEMORY ON UNIBUS TEST

* THIS TEST DETERMINES IF IT HAS BEEN SELECTED BY A '1' IN BIT 5 OF
 * THE SWITCH REGISTER. IF IT HAS, IT THEN DETERMINES IF THERE IS ANY
 * UNIBUS MEMORY - AN ERROR RESULTS IF THERE IS NONE. IF THERE IS MEMORY,
 * IT THEN SIZES THE AMOUNT OF MEMORY ON THE UNIBUS, INFORMS THE USER
 * HOW MUCH MEMORY IT FOUND ON THE FIRST PASS, THEN SETS AND CLEARS ALL
 * BITS OF ALL LOCATIONS IN THE UNIBUS MEMORY FOUND USING THE 'MARCH'
 * ALGORITHM.

M7098 FOR THE 11/44, M7134 FOR THE 11/24
 JUMPER SETTINGS FOR THE MAP REGISTERS



*W3-W7 AND W8-W12 ARE THE BINARY-CODED PAGE NUMBER LIMIT (UPPER OR LOWER).
 *A JUMPER IN CORRESPONDS TO A LOGIC '0'; A JUMPER OUT TO A LOGIC '1'. TO
 *SET THE JUMPERS, DETERMINE WHICH UNIBUS PAGE THE MEMORY RESIDES IN (0 TO 31)
 *BY CHECKING WHICH OF THE 5 ADDRESS BITS BA17-BA13 ARE ASSERTED. ALL ZEROS
 *IS PAGE 0, 10000 IS PAGE 1, 01000 IS PAGE 2, 11000 IS PAGE 3, ETC. UP TO
 *PAGE 31 (11111). FOR THE MEMORY TO BE DETECTED, IT MUST LIE AT OR ABOVE
 *THE LOWER LIMIT JUMPER SETTING AND BELOW THE UPPER LIMIT JUMPER SETTING.
 *THUS TO HAVE UNIBUS MEMORY IN PAGES 5-9, ONE WOULD SET THE LOWER LIMIT TO
 *PAGE 5 (10100) OR W10 AND W12 OUT, AND W8, W9 AND W11 IN, AND THE UPPER LIMIT
 *TO PAGE 10 (01010) OR W4 AND W6 OUT, AND W3, W5 AND W7 IN. UNIBUS MEMORY
 *MUST BE CONTIGUOUS SINCE NO GAPS ARE PERMITTED.

2752	017202	000004	000001	177572
2753	017204	042737	000040	161730
2754	017212	032777		
2755	017220	001417		
2756	017222	012737	021306	001346
2757	017230	022737	170000	001254
2758	017236	001012		
2759	017240	022737	177400	001256
2760	017246	001006		
2761	017250	012737	012620	001106
2762	017256	104017		
2763	017260	000137	021306	
2764	017264	005037	001302	
2765	017270	005037	001314	
2766	017274	005037	172516	
2767	017300	052737	000020	172516
	017306	005037	172340	

```

*****
TST31: SCOPE
BIC #1,MMRO ;TURN OFF MEMORY MANAGEMENT
BIT #BIT5,@SWR ;SEE IF THIS TEST HAS BEEN SELECTED
BEQ 2$ ;BRANCH TO JUMP IF TEST NOT SELECTED
MOV #SEOP,NXTTST ;POINT TO ESCAPE VECTOR
1$: CMP #170000,LOWEST ;SEE IF LOWEST IS LOWEST
BNE 3$ ;GO DO TEST IF IT ISN'T - UNIBUS MEMORY EXISTS
CMP #177400,HIGEST ;SEE IF HIGEST IS HIGEST
BNE 3$ ;GO DO TEST IF IT ISN'T - UNIBUS MEMORY EXISTS
MOV #SIZEJ,$LPERR ;MOVE SIZE JUMPER ROUTINE TO LOOP ON ERROR
ERROR +17 ;NO UNIBUS MEMORY EXISTS
2$: JMP $EOP ;JUMP TO END OF PASS
3$: CLR ERRCNT ;CLEAR THE ERROR COUNT INDICATOR
CLR PCPUER ;CLEAR THE ERROR REGISTER RECEIVER
CLR MMR3 ;CLEAR MEMORY MANAGEMENT REGISTER MMR3
BIS #20,MMR3 ;TURN ON 22-BIT MAPPING
CLR KIPARO ;MAP PARO TO 0-4K
    
```

2768 017312 012737 000200 172342
2769 017320 013700 001272
2770 017324 163700 001270
2771 017330 005200
2772 017332 010001
2773 017334 005046

4\$:

MOV #200,KIPAR1
MOV UBRHI,RO
SUB UBRLOW,RO
INC RO
MOV RO,R1
CLR -(SP)

J 10

;MAP PAR1 TO 4-8K
;MOVE UBRHI TO RO
;SUBTRACT UBRLOW FROM IT, AND
;ADD LAST BLOCK OF 4K TO LOOP COUNTER
;SAVE RO IN R1
;CLEAR THE MAJOR LOOP INDICATOR ON STACK

SEQ 0126

2774	017336	012746	000200		MOV	#200,-(SP)	;MOVE PAR CHANGE TO STACK	
2775	017342	013746	001260		MOV	UBMLOW,-(SP)	;MOVE STARTING PAR VALUE TO STACK	
2776	017346	012746	000002		MOV	#2,-(SP)	;MOVE INCREMENT VALUE TO STACK	
2777	017352	012704	125252		MOV	#125252,R4	;MOVE FIRST TEST PATTERN TO R4	
2778	017356	012705	052525		MOV	#52525,R5	;MOVE SECOND TEST PATTERN TO R5	
2779	017362	005737	020022	5\$:	TST	\$PASS	;SEE IF THIS IS FIRST PASS	
2780	017366	001015			BNE	6\$;BRANCH IF NOT	
2781	017370	010046			MOV	R0,-(SP)	;MOVE LOOP COUNTER TO THE STACK AND	
2782	017372	006316			ASL	(SP)	;ROTATE THIS TO THE LEFT 2 PLACES	
2783	017374	006316			ASL	(SP)	;TO INDICATE NUMBER OF K IN OCTAL	
2784	017376	112737	000001	002413	MOVB	#1,SPSUPP	;MOVE THE SUPPRESS SPACES INDICATOR TO FLAG	
2785	017404	104400	021735		TYPE	,UBMAVA	;GO TYPE THE UNI-BUS MEMORY AVAILABLE MESSAGE	
2786	017410	104410			TYPDS		;GO TYPE THE NUMBER IN DECIMAL	
2787	017412	104400	021771		TYPE	,UBMEND	;TYPE A " K" AND <CRLF>	
2788	017416	105037	002413		CLRB	SPSUPP	;CLEAR THE INDICATOR FLAG	
2789	017422	013737	001260	172354	6\$:	MOV	UBMLOW,KIPAR6	;INITIALIZE PAR6
2790	017430	010100			MOV	R1,R0	;REINITIALIZE LOOP COUNTER R0	
2791	017432	052737	000001	177572		BIS	#1,MMR0	;TURN ON MEMORY MANAGEMENT
2792	017440	012702	010000	7\$:	MOV	#10000,R2	;ACCESS ALL WORDS IN THIS 4K BLOCK	
2793	017444	012703	140000		MOV	#140000,R3	;FIRST ADDRESS OF THIS PAGE	
2794	017450	010423		8\$:	MOV	R4,(R3)+	;MOVE THE PATTERN TO THE LOCATION	
2795	017452	077202			SOB	R2,8\$;SUBTRACT 1 AND BRANCH IF 4K NOT DONE	
2796	017454	062737	000200	172354		ADD	#200,KIPAR6	;MAP TO NEXT 4K BLOCK
2797	017462	077012			SOB	R0,7\$;SUBTRACT 1 AND BRANCH IF BLOCKS OF 4K NOT DONE	
2798	017464	016637	000002	172354		MOV	2(SP),KIPAR6	;REINITIALIZE KIPAR6 TO POINT AT BEGINNING
2799	017472	010100			MOV	R1,R0	;REINITIALIZE LOOP COUNTER R0	
2800	017474	012702	010000	9\$:	MOV	#10000,R2	;ACCESS ALL WORDS IN THIS 4K BLOCK	
2801	017500	012703	140000		MOV	#140000,R3	;FIRST ADDRESS OF THIS PAGE	
2802	017504	066603	000006		ADD	6(SP),R3	;ADD OFFSET FOR THIS MAJOR PASS	
2803	017510	012737	017520	001106		MOV	#20\$,\$LPERR	;LOOP ON ERROR TO 20\$
2804	017516	000401			BR	10\$;BRANCH OVER LOOP ON ERROR PREPARATION	
2805	017520	010413		20\$:	MOV	R4,(R3)	;REWRITE 1ST PATTERN TO LOCATION FOR LOOP	
2806	017522	020413		10\$:	CMP	R4,(R3)	;SEE IF IT WAS LOADED PROPERLY	
2807	017524	001403			BEQ	11\$;BRANCH AROUND ERROR CALL IF OK	
2808	017526	010437	001202		MOV	R4,\$TMP4	;MOVE EXPECTED DATA TO \$TMP4	
2809	017532	000405			BR	12\$;GO COMPLETE DATA FETCHING AND CALL ERROR	
2810	017534	005113		11\$:	COM	(R3)	;COMPLEMENT THAT LOCATION TO PRODUCE SECOND TEST PATTERN	
2811	017536	020513			CMP	R5,(R3)	;SEE IF IT IS THE COMPLEMENT	
2812	017540	001417			BEQ	16\$;BRANCH AROUND ERROR CALL IF IT IS	
2813	017542	010537	001202		MOV	R5,\$TMP4	;MOVE EXPECTED DATA TO \$TMP4	
2814	017546	011337	001204	12\$:	MOV	(R3),\$TMP5	;MOVE RECEIVED DATA TO \$TMP5	
2815	017552	005737	001314		TST	PCPUER	;SEE IF THIS ACCESS TIMED OUT - IF IT DID, BRANCH	
2816	017556	001006			BNE	15\$;AROUND ERROR CALLS - TIMEOUT ROUTINE LOGGED ERROR	
2817	017560	010337	006144		MOV	R3,EADRES	;MOVE ADDRESS IN R3 TO EADRES FOR ERROR CALL	
2818	017564	104206		13\$:	ERROR	+206	;DATA PATTERN NOT CORRECT	
2819	017566	000404			BR	16\$;BRANCH AROUND INCREMENT AND CLEAR INSTRUCTIONS	
2820	017570	005237	001110	14\$:	INC	\$ERTTL	;STILL COUNT THIS AS AN ERROR	
2821	017574	005037	001314	15\$:	CLR	PCPUER	;CLEAR TIMEOUT RECEIVER	
2822	017600	061603		16\$:	ADD	(SP),R3	;ADD INCREMENT/DECREMENT VALUE TO R3	
2823	017602	077231			SOB	R2,10\$;SUBTRACT 1 AND BRANCH IF 4K NOT CHECKED	
2824	017604	066637	000004	172354		ADD	4(SP),KIPAR6	;MAP TO NEXT 4K BLOCK

2831	017640	012766	177600	000004	MOV	#-200,4(SP)	;MOVE REVERSE PAR STEP TO STACK
2832	017646	013766	001262	000002	MOV	UBMHI,2(SP)	;MOVE LAST PAR VALUE TO STACK
2833	017654	012716	177776		MOV	#-2,(SP)	;MOVE DECREMENT VALUE TO STACK
2834	017660	000660			BR	6\$;BRANCH BACK FOR SECOND PASS

```
2835 .SBTTL PRETEST DATA SETUP SUBROUTINE
2836 :*****
2837 017662 013777 001100 161262 PRETST: MCV $STNM,@DISPLAY ;DISPLAY TEST NUMBER FOR ALL TO SEE
2838 017670 010046 MOV RO,-(SP) ;SAVE RO
2839 017672 016600 000002 MOV 2(SP),RO ;MOVE ORIGINAL RETURN ADDRESS TO RO
2840 017676 012037 001346 MOV (RO)+,NXTTST ;SAVE STARTING ADDRESS OF NEXT TEST FOR ESCAPE ON PAR ERRORS
2841 017702 012037 001106 MOV (RO)+,$LPERR ;SET LOOP ON ERROR POINTER TO 20$
2842 017706 012037 001100 MOV (RO)+,$STNM ;SETUP TEST NUMBER AND CLEAR THE ERROR FLAG
2843 017712 010066 000002 MOV RO,2(SP) ;FUDGE RETURN OVER DATA
2844 017716 012600 MOV (SP)+,RO ;RESTORE RO
2845 017720 011637 001104 MOV (SP),$LPADR ;SET LOOP ON TEST POINTER TO START OF TEST
2846 017724 000207 RTS PC ;RETURN TO TEST
```

PRETEST DATA SETUP SUBROUTINE

```
2847      020000      . = 20000      ; THE APT TABLES NEED TO START AT 20000 - THIS STATEMENT DOES THAT
2848
2849      177777      ADDW0 = 177777
2850      177777      ADDW1 = 177777
```

2851

000024 020000
000024 000024
000044 000200
000044 000044
000044 020000
020000 020000
020000 000000
020002 020014
020004 000005
020006 000010
020010 000000
020012 000052

```
.SBTTL APT PARAMETER BLOCK
:*****
:SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
:*****
      .SX=.      ;;SAVE CURRENT LOCATION
      .=24      ;;SET POWER FAIL TO POINT TO START OF PROGRAM
      200       ;;FOR APT START UP
      .=44      ;;POINT TO APT INDIRECT ADDRESS PNTR.
      $APTHDR   ;;POINT TO APT HEADER BLOCK
      .=.SX     ;;RESET LOCATION COUNTER
:*****
:SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
:INTERFACE SPEC.
$APTHD:
$HIBTS: .WORD 0      ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
$MBADR: .WORD $MAIL  ;;ADDRESS OF APT MAILBOX (BITS 0-15)
$TSTM:  .WORD 5      ;;RUN TIM OF LONGEST TEST
$PASTM: .WORD 10     ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
$UNITM: .WORD 0      ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
      .WORD $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
```

2853

.SBTTL APT MAILBOX-ETABLE

::*****

```
.EVEN
020014 $MAIL:                ;;APT MAILBOX
020014 000000 $MSGTY: .WORD   AMSGTY  ;;MESSAGE TYPE CODE
020016 000000 $FATAL: .WORD   AFATAL  ;;FATAL ERROR NUMBER
020020 000000 $TESTN: .WORD   ATESTN  ;;TEST NUMBER
020022 000000 $PASS:  .WORD   APASS   ;;PASS COUNT
020024 000000 $DEVCT: .WORD   ADEVCT  ;;DEVICE COUNT
020026 000000 $UNIT:  .WORD   AUNIT   ;;I/O UNIT NUMBER
020030 000000 $MSGAD: .WORD   AMSGAD  ;;MESSAGE ADDRESS
020032 000000 $MSGLG: .WORD   AMSGLG  ;;MESSAGE LENGTH
020034 $ETABLE:           ;;APT ENVIRONMENT TABLE
020034    000 $ENV: .BYTE   AENV   ;;ENVIRONMENT BYTE
020035    000 $ENVM: .BYTE   AENVM  ;;ENVIRONMENT MODE BITS
020036 000000 $SWREG: .WORD   ASWREG  ;;APT SWITCH REGISTER
020040 000000 $USWR:  .WORD   AUSWR   ;;USER SWITCHES
020042 000000 $CPUOP: .WORD   ACPUOP  ;;CPU TYPE,OPTIONS
::*
::*          BITS 15-11=CPU TYPE
::*          11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
::*          11/70=06,PDQ=07,Q=10
::*
::*          BIT 10=REAL TIME CLOCK
::*          BIT 9=FLOATING POINT PROCESSOR
::*          BIT 8=MEMORY MANAGEMENT
020044    000 $MAMS1: .BYTE   AMAMS1  ;;HIGH ADDRESS,M.S. BYTE
020045    000 $MTYP1: .BYTE   AMTYP1 ;;MEM. TYPE,BLK#1
::*
::*          MEM.TYPE BYTE -- (HIGH BYTE)
::*          900 NSEC CORE=001
::*          300 NSEC BIPOLAR=002
::*          500 NSEC MOS=003
020046 000000 $MADR1: .WORD   AMADR1  ;;HIGH ADDRESS,BLK#1
::*          MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
020050    000 $MAMS2: .BYTE   AMAMS2  ;;HIGH ADDRESS,M.S. BYTE
020051    000 $MTYP2: .BYTE   AMTYP2  ;;MEM.TYPE,BLK#2
020052 000000 $MADR2: .WORD   AMADR2  ;;MEM.LAST ADDRESS,BLK#2
020054    000 $MAMS3: .BYTE   AMAMS3  ;;HIGH ADDRESS,M.S.BYTE
020055    000 $MTYP3: .BYTE   AMTYP3  ;;MEM.TYPE,BLK#3
020056 000000 $MADR3: .WORD   AMADR3  ;;MEM.LAST ADDRESS,BLK#3
020060    000 $MAMS4: .BYTE   AMAMS4  ;;HIGH ADDRESS,M.S.BYTE
020061    000 $MTYP4: .BYTE   AMTYP4  ;;MEM.TYPE,BLK#4
020062 000000 $MADR4: .WORD   AMADR4  ;;MEM.LAST ADDRESS,BLK#4
020064 000000 $VECT1: .WORD   AVECT1  ;;INTERRUPT VECTOR#1,BUS PRIORITY#1
020066 000000 $VECT2: .WORD   AVECT2  ;;INTERRUPT VECTOR#2BUS PRIORITY#2
020070 000000 $BASE:  .WORD   ABASE   ;;BASE ADDRESS OF EQUIPMENT UNDER TEST
020072 000000 $DEVN:  .WORD   ADEVN   ;;DEVICE MAP
020074 000000 $CDW1:  .WORD   ACDW1   ;;CONTROLLER DESCRIPTION WORD#1
020076 000000 $CDW2:  .WORD   ACDW2   ;;CONTROLLER DESCRIPTION WORD#2
020100 177777 $DDW0:  .WORD   ADDW0   ;;DEVICE DESCRIPTOR WORD#0
020102 177777 $DDW1:  .WORD   ADDW1   ;;DEVICE DESCRIPTOR WORD#1
020104 000000 $DDW2:  .WORD   ADDW2   ;;DEVICE DESCRIPTOR WORD#2
020106 000000 $DDW3:  .WORD   ADDW3   ;;DEVICE DESCRIPTOR WORD#3
```

020110 000000
020112 000000
020114 000000
020116 000000
020120 000000
020122 000000

E 11
\$DDW4: .WORD ADDW4 ;;DEVICE DESCRIPTOR WORD#4
\$DDW5: .WORD ADDW5 ;;DEVICE DESCRIPTOR WORD#5
\$DDW6: .WORD ADDW6 ;;DEVICE DESCRIPTOR WORD#6
\$DDW7: .WORD ADDW7 ;;DEVICE DESCRIPTOR WORD#7
\$DDW8: .WORD ADDW8 ;;DEVICE DESCRIPTOR WORD#8
\$DDW9: .WORD ADDW9 ;;DEVICE DESCRIPTOR WORD#9

SEQ 0134

020124	000000	\$DDW10:	.WORD	ADDW10	::DEVICE	DESCRIPTOR	WORD#10
020126	000000	\$DDW11:	.WORD	ADDW11	::DEVICE	DESCRIPTOR	WORD#11
020130	000000	\$DDW12:	.WORD	ADDW12	::DEVICE	DESCRIPTOR	WORD#12
020132	000000	\$DDW13:	.WORD	ADDW13	::DEVICE	DESCRIPTOR	WORD#13
020134	000000	\$DDW14:	.WORD	ADDW14	::DEVICE	DESCRIPTOR	WORD#14
020136	000000	\$DDW15:	.WORD	ADDW15	::DEVICE	DESCRIPTOR	WORD#15
020140		\$ETEND:					

2855

.SBTTL SCOPE HANDLER ROUTINE

```

*****
*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
*SW14=1      LOOP ON TEST
*SW11=1      INHIBIT ITERATIONS
*SW09=1      LOOP ON ERROR
*SW08=1      LOOP ON TEST IN SWR<4:0>
*CALL
*

```

* SCOPE ;:SCOPE=IOT

```

$SCOPE:
020140 005037 001344      CLR      RETRY      ;CLEAR RETRY FLAG AN THE START OF EACH TEST
020144 005037 001302      CLR      ERRCNT     ;CLEAR THE MULTIPLE ERROR COUNTER
020150 005037 001244      CLR      DATAOR    ;LOCATION FOR LOGICAL OR OF BAD DATA
020154 005037 001234      CLR      ADDROR     ;LOCATION FOR LOGICAL OR OF ADDRESS
020160 005037 001236      CLR      ADDROR+2    ;LOCATION FOR UPPER 6 BITS OF LOGICAL OR OF ADDRESS
020164 005037 001252      CLR      PATTOR     ;LOCATION FOR LOGICAL OR OF PATTERN LOADED
020170 012737 177777 001240  MOV      #-1,DATAND  ;LOCATION FOR LOGICAL AND OF BAD DATA
020176 012737 177777 001230  MOV      #-1,ADRAND  ;LOCATION FOR LOGICAL AND OF ADDRESS
020204 012737 000077 001232  MOV      #77,ADRAND+2 ;LOCATION FOR UPPER 6 BITS OF LOGICAL AND OF ADDRESS
020212 012737 177777 001250  MOV      #-1,PATAND  ;LOCATION FOR LOGICAL AND OF PATTERN LOADED
020220 012737 000077 006146  MOV      #77,EADRES+2 ;RESTORE UPPER 6 BIT LOCATION OF EADRES+2
020226 012737 000077 006152  MOV      #77,EADRS2+2 ;RESTORE UPPER 6 BIT LOCATION OF EADRS2+2
020234 032777 040000 160706  BIT      #40000,@SWR ;&& LOOP ON PRESENT TEST?
020242 001117      BNE      $OVER      ;&& YES IF SW14=1

;#####START OF CODE FOR THE XOR TESTER#####
020244 000416      $XTSTR: BR      6$      ;:IF RUNNING ON THE 'XOR' TESTER CHANGE
                                ;:THIS INSTRUCTION TO A 'NOP' (NOP=240)
020246 013746 000004      MOV      @#ERRVEC,-(SP) ;:SAVE THE CONTENTS OF THE ERROR VECTOR
020252 012737 020272 000004  MOV      #5$,@#ERRVEC  ;:SET FOR TIMEOUT
020260 005737 177060      TST      @#177060      ;:TIME OUT ON XOR?
020264 012637 000C04      MOV      (SP)+,@#ERRVEC ;:RESTORE THE ERROR VECTOR
020270 000466      BR      $$VLAD        ;:GO TO THE NEXT TEST
020272 022626      5$: CMP      (SP)+,(SP)+   ;:CLEAR THE STACK AFTER A TIME OUT
020274 012637 000004      MOV      (SP)+,@#ERRVEC ;:RESTORE THE ERROR VECTOR
020300 000426      BR      7$          ;:LOOP ON THE PRESENT TEST
020302      6$:;#####END OF CODE FOR THE XOR TESTER#####
020302 032777 000400 160640  BIT      #BIT08,@SWR   ;&& LOOP ON SPEC. TEST?
020310 001407      BEQ      2$          ;:BR IF NO
020312 017746 160632      MOV      @SWR,-(SP)    ;&& SET DESIRED TEST NUM. FROM SWR
020316 042716 000340      BIC      #$$SWRMK,(SP) ;:STRIP AWAY UNDESIRED BITS
020322 122637 001100      CMPB     (SP)+,$TSTNM  ;:ON THE RIGHT TEST?
020326 001465      BEQ      $OVER        ;:BR IF YES
020330 105737 001101      2$: TSTB     $ERFLG      ;:HAS AN ERROR OCCURRED?
020334 001421      BEQ      3$          ;:BR IF NO
020336 123737 001113 001101  CMPB     $ERMAX,$ERFLG ;:MAX. ERRORS FOR THIS TEST OCCURRED?
020344 101015      BHI      3$          ;:BR IF NO
020346 032777 001000 160574  BIT      #BIT09,@SWR   ;&& LOOP ON ERROR?
020354 001404      BEQ      4$          ;:BR IF NO

```

020356	013737	001106	001104	7\$:	MOV	\$LPERR,\$LPADR	::SET LOOP ADDRESS TO LAST SCOPE
020364	000446				BR	\$OVER	
020366	105037	001101		4\$:	CLRB	\$ERFLG	::ZERO THE ERROR FLAG
020372	005037	001210			CLR	\$TIMES	::CLEAR THE NUMBER OF ITERATIONS TO MAKE
020376	000415				BR	1\$::ESCAPE TO THE NEXT TEST
020400	032777	004000	160542	3\$:	BIT	#BIT11,@SWR	::INHIBIT ITERATIONS?

H 11

SEQ 0137

020406	001011			BNE	1\$::BR IF YES
020410	005737	020022		TST	\$PASS		::IF FIRST PASS OF PROGRAM
020414	001406			BEQ	1\$:: INHIBIT ITERATIONS
020416	005237	001102		INC	\$ICNT		::INCREMENT ITERATION COUNT
020422	023737	001210	001102	CMP	\$TIMES,\$ICNT		::CHECK THE NUMBER OF ITERATIONS MADE
020430	002024			BGE	\$OVER		::BR IF MORE ITERATION REQUIRED
020432	012737	000001	001102	1\$: MOV	#1,\$ICNT		::REINITIALIZE THE ITERATION COUNTER
020440	013737	020516	001210	MOV	\$MXCNT,\$TIMES		::SET NUMBER OF ITERATIONS TO DO
020446	105237	001100		\$SVLAD: INCB	\$TSTNM		::COUNT TEST NUMBERS
020452	113737	001100	020020	MOVB	\$TSTNM,\$TESTN		::&& SET TEST # IN APT MAILBOX
020460	011637	001104		MOV	(SP),\$LPADR		::SAVE SCOPE LOOP ADDRESS
020464	011637	001106		MOV	(SP),\$LPERR		::SAVE ERROR LOOP ADDRESS
020470	005037	001212		CLR	\$ESCAPE		::CLEAR THE ESCAPE FROM ERROR ADDRESS
020474	112737	000001	001113	MOVB	#1,\$ERMAX		::ONLY ALLOW ONE(1) ERROR ON NEXT TEST
020502	013777	001100	160442	\$OVER: MOV	\$TSTNM,@DISPLAY		::&& DISPLAY TEST NUMBER
020510	013716	001104		MOV	\$LPADR,(SP)		::FUDGE RETURN ADDRESS
020514	000002			RTI			::FIXES PS
020516	000002			\$MXCNT: 2.			::MAX. NUMBER OF ITERATIONS

2857

```

.SBTTL ERROR HANDLER ROUTINE
;*****
;THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
;SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
;AND GO TO ERTYPE ON ERROR
;THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
;SW15=1      HALT ON ERROR
;           HALT CAN OCCUR BEFORE AND AFTER THE ERROR TYPEOUT
;SW13=1      INHIBIT ERROR TYPEOUTS
;SW10=1      BELL ON ERROR
;SW09=1      LOOP ON ERROR
;CALL
;           ERROR      N      ;;ERROR=EMT AND N=ERROR ITEM NUMBER
$ERROR:
020520      113737  001100  001310      MOVB  $STNM,TESTNO  ;SAVE TEST NUMBER FOR ERROR TYPE OUT
020526      005237  001302                INC  ERRCNT         ;COUNT ALL MULTIPLE ERRORS
020532      010037  001156                MOV  R0,$REG0       ;SAVE R0 FOR POSSIBLE TYPE OUT
020536      010137  001160                MOV  R1,$REG1       ;SAVE R1 FOR POSSIBLE TYPE OUT
020542      010237  001162                MOV  R2,$REG2       ;SAVE R2 FOR POSSIBLE TYPE OUT
020546      010337  001164                MOV  R3,$REG3       ;SAVE R3 FOR POSSIBLE TYPE OUT
020552      010437  001166                MOV  R4,$REG4       ;SAVE R4 FOR POSSIBLE TYPE OUT
020556      010537  001170                MOV  R5,$REG5       ;SAVE R5 FOR POSSIBLE TYPE OUT
020562      105237  001101      7$:      INCB  $ERFLG        ;SET THE ERROR FLAG
020566      001775                BEQ  7$             ;DON'T LET THE FLAG GO TO ZERO
020570      013777  001100  160354      MOV  $STNM,@DISPLAY ;&& DISPLAY TEST NUMBER AND ERROR FLAG
;TST  @SWR                ;&& HALT ON ERROR = 1?\
;BPL  8$                ;BRANCH IF NO      >(REMOVED FOR CKKUAB)
;HALT                    ;YES--HALT /
020576      032777  002000  160344      8$:      BIT   #BIT10,@SWR   ;&& BELL ON ERROR?
020604      001402                BEQ  1$             ;NO - SKIP
020606      104400  001214                TYPE ,$BELL        ;RING BELL
020612      005237  001110      1$:      INC  $ERTTL        ;COUNT THE NUMBER OF ERRORS
020616      011637  001114                MOV  (SP),$ERRPC   ;GET ADDRESS OF ERROR INSTRUCTION
020622      162737  000002  001114      SUB  #2,$ERRPC
020630      117737  160260  001112      MOVB @$ERRPC,$ITEMB ;STRIP AND SAVE THE ERROR ITEM CODE
020636      032777  020000  160304      BIT  #BIT13,@SWR   ;&& SKIP TYPEOUT IF SET
020644      001017                BNE  2$             ;SKIP TYPEOUTS
020646      004737  002012                JSR  PC,ERTYPE     ;GO TO USER ERROR ROUTINE
020652      104400  001221                TYPE ,$CRLF
020656      122737  000001  020034      CMPB #1,$ENV       ;&& IS THIS APT?
020664      001007                BNE  2$             ;&& NO
020666      113737  001112  020700      MOVB $ITEMB,21$   ;&&
020674      004737  021056                JSR  PC,$ATY4      ;&&
020700      000                21$:      .BYTE 0            ;&&
020701      000                .BYTE 0            ;&&
020702      000777                22$:      BR   22$           ;&&
020704      005777  160240      2$:      TST  @SWR         ;&& HALT ON ERROR
020710      100007                BPL  9$             ;SKIP IF CONTINUE
020712      032777  000200  160174      BIT  #BIT7,@$ERRPC ;&& LOOK AT ERROR AND SEE IF A +2XX ERROR
020720      001402                BEQ  25$           ;&& BRANCH IF NOT TO HALT
020722      005037  001302                CLR  ERRCNT        ;&& CLEAR ERROR COUNT SO RESTART PRINTS ANOTHER HEADER

```

020726	000000			25\$:	HALT							
020730	022737	021534	000042	9\$:	CMP	#SENDAD,42			:::HALT ON ERROR!			
020736	001001				BNE	3\$:::ACT-11?			SEQ 0140
020740	000000				HALT				:::BRANCH IF NO			
020742	032777	001000	160200	3\$:	BIT	#BIT09,@SWR			:::YES			
020750	001402				BEQ	4\$:::LOOP ON ERROR SWITCH SET?			
									:::BR IF NO			

K 11

```
020752 013716 001106          MOV    $LPERR,(SP)      ;;FUDGE RETURN FOR LOOPING
020756 005737 001212      4$:   TST    $ESCAPE        ;;CHECK FOR AN ESCAPE ADDRESS
020762 001402              BEQ    5$              ;;BR IF NONE
020764 013716 001212          MOV    $ESCAPE,(SP)    ;;FUDGE RETURN ADDRESS FOR ESCAPE
020770              5$:
020770 032777 001000 160152    BIT    #SW9,@SWR        ;ARE WE LOOPING ON THIS ERROR?
020776 001417              BEQ    1000$           ;BRANCH IF NOT
021000 012737 177777 177766    MOV    #-1,CPUERR      ;CLEAR CPU ERROR REGISTER
021006 042737 177776 177572    BIC    #177776,MMRO    ;CLEAR MEMORY MANAGEMENT STATUS REGISTER
021014 012737 177777 004412    MOV    #-1,TOFLAG     ;INITIALIZE TRAP FLAG
021022 012737 177777 004604    MOV    #-1,CPFLAG     ;INITIALIZE CP TRAP FLAG
021030 012737 177777 004746    MOV    #-1,MMFLAG     ;INITIALIZE MEMORY MANAGEMENT TRAP FLAG
021036 000002      1000$: RTI          ;RETURN TO TEST
```

2859

.SBTTL APT COMMUNICATIONS ROUTINE

```

*****
021040 112737 000001 021304 $ATY1: MOVB #1,$FFLG ;;TO REPORT FATAL ERROR
021046 112737 000001 021302 $ATY3: MOVB #1,$MFLG ;;TO TYPE A MESSAGE
021054 000403
021056 112737 000001 021304 $ATY4: MOVB #1,$FFLG ;;TO ONLY REPORT FATAL ERROR
021064 $ATYC:
021064 010046 MOV R0,-(SP) ;;PUSH R0 ON STACK
021066 010146 MOV R1,-(SP) ;;PUSH R1 ON STACK
021070 105737 021302 TSTB $MFLG ;;SHOULD TYPE A MESSAGE?
021074 001450 BEQ 5$ ;;IF NOT: BR
021076 122737 000001 020034 CMPB #APTENV,$ENV ;;OPERATING UNDER APT?
021104 001031 BNE 3$ ;;IF NOT: BR
021106 132737 000100 020035 BITB #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
021114 001425 BEQ 3$ ;;IF NOT: BR
021116 017600 000004 MOV @4(SP),R0 ;;GET MESSAGE ADDR.
021122 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDR.
021130 005737 020014 1$: TST $MSGTYPE ;;SEE IF DONE W/ LAST XMISSION?
021134 001375 BNE 1$ ;;IF NOT: WAIT
021136 010037 020030 MOV R0,$MSGAD ;;PUT ADDR IN MAILBOX
021142 105720 2$: TSTB (R0)+ ;;FIND END OF MESSAGE
021144 001376 BNE 2$
021146 163700 020030 SUB $MSGAD,R0 ;;SUB START OF MESSAGE
021152 006200 ASR R0 ;;GET MESSAGE LNGTH IN WORDS
021154 010037 020032 MOV R0,$MSGGLT ;;PUT LENGTH IN MAILBOX
021160 012737 000004 020014 MOV #4,$MSGTYPE ;;TELL APT TO TAKE MSG.
021166 000413 BR 5$
021170 017637 000004 021214 3$: MOV @4(SP),4$ ;;PUT MSG ADDR IN JSR LINKAGE
021176 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDRESS
021204 013746 177776 MOV 177776,-(SP) ;;PUSH 177776 ON STACK
021210 004737 002674 JSR PC,$TYPE ;;CALL TYPE MACRO
021214 000000 4$: .WORD 0
021216 5$:
021216 105737 021304 10$: TSTB $FFLG ;;SHOULD REPORT FATAL ERROR?
021222 001416 BEQ 12$ ;;IF NOT: BR
021224 005737 020034 TST $ENV ;;RUNNING UNDER APT?
021230 001413 BEQ 12$ ;;IF NOT: BR
021232 005737 020014 11$: TST $MSGTYPE ;;FINISHED LAST MESSAGE?
021236 001375 BNE 11$ ;;IF NOT: WAIT
021240 017637 000004 020016 MOV @4(SP),$FATAL ;;GET ERROR #
021246 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDR.
021254 005237 020014 INC $MSGTYPE ;;TELL APT TO TAKE ERROR
021260 105037 021304 12$: CLRB $FFLG ;;CLEAR FATAL FLAG
021264 105037 021303 CLRB $LFLG ;;CLEAR LOG FLAG
021270 105037 021302 CLRB $MFLG ;;CLEAR MESSAGE FLAG
021274 012601 MOV (SP)+,R1 ;;POP STACK INTO R1
021276 012600 MOV (SP)+,R0 ;;POP STACK INTO R0
021300 000207 RTS PC ;;RETURN
021302 000 $MFLG: .BYTE 0 ;;MESSG. FLAG
021303 000 $LFLG: .BYTE 0 ;;LOG FLAG
021304 000 $FFLG: .BYTE 0 ;;FATAL FLAG
                .EVEN
000200 APTSIZE=200
000001 APTENV=001
000100 APTSPOOL=100
000040 APTCSUP=040
    
```

2861

.SBTTL END OF PASS ROUTINE

```

*****
*INCREMENT THE PASS NUMBER ($PASS)
*INDICATE END-OF-PROGRAM AFTER 1 PASSES THRU THE PROGRAM
*TYPE "END PASS #XXXXX TOTAL NUMBER OF ERRORS SINCE LAST REPORT YYYYYY"
*WHERE XXXXX AND YYYYYY ARE DECIMAL NUMBERS
*IF SW12=1 INHIBIT TRACE TRAP
*IF THERES A MONITOR GO TO IT
*IF THERE ISN'T JUMP TO LOOP

```

```

$EOP:
021306                                SCOPE                                ;LOOP ON LAST TEST
021306 000004                          CLR      MMRO                        ;TURN OFF FULL RELOCATION
021310 005037 177572                    CLR      MMR3                        ;DISABLE THE UNIBUS MAP
021314 005037 172516                    TBTR                                ;RESTORE THE T BIT IF IT WAS ON
021320 104420                            CLR      $TSTNM                       ;;ZERO THE TEST NUMBER
021322 005037 001100                    CLR      $TIMES                       ;;ZERO THE NUMBER OF ITERATIONS
021326 005037 001210                    INC      $PASS                        ;;INCREMENT THE PASS NUMBER
021332 005237 020022                    BIC      #100000,$PASS                ;;DON'T ALLOW A NEG. NUMBER
021336 042737 100000 020022            DEC      (PC)+                        ;;LOOP?
021344 005327                            $EOPCT: .WORD 1
021346 000001                            BGT      $DOAGN                       ;;YES
021350 003075                            MOV      (PC)+,@(PC)+                 ;;RESTORE COUNTER
021352 012737                            $ENDCT: .WORD 1
021354 000001                            $EOPCT
021356 021346                            $NEXT:
021360                                TYPE      ,65$                        ;;TYPE ASCIZ STRING
021360 104400 021366                    BR       64$                          ;;GET OVER THE ASCIZ
021364 000407                            .ASCIZ  <15><12>/END PASS #/
021366   015   012   105 65$:
.EVEN
64$:
021404                                MOV      $PASS,-(SP)                  ;;SAVE $PASS FOR TYPEOUT
021404 013746 020022                    ;;TYPE PASS NUMBER
021410                                TYPDS
021410 104410                            TST     $ERTTL                        ;;GO TYPE--DECIMAL ASCII WITH SIGN
021412 005737 001110                    BEQ     70$                          ;;SEE IF THERE WERE ANY ERRORS THIS PASS
021416 001427                            TYPE    ,67$                          ;;BRANCH OVER ERROR REPORT IF NONE
021420 104400 021426                    BR      66$                          ;;TYPE ASCIZ STRING
021424 000421                            .ASCIZ  / TOTAL ERRORS SINCE LAST REPORT /
021426   040   040   124 67$:
.EVEN
66$:
021470                                MOV      $ERTTL,-(SP)                 ;;SAVE $ERTTL FOR TYPEOUT
021470 013746 001110                    ;;TOTAL NUMBER OF ERRORS
021474 104410                            TYPDS
021474 104400 001221                    TYPE    ,%CRLF                       ;;GO TYPE--DECIMAL ASCII WITH SIGN
021502 005037 001110                    CLR     $ERTTL                        ;;TYPE CARRIAGE RETURN, LINE FEED
021506 013700 000042                    $GET42: MOV  @#42,R0                    ;;CLEAR ERROR TOTAL
021512 001414                            ;;GET MONITOR ADDRESS
021514 005046                            BEQ     $DOAGN                        ;;BRANCH IF NO MONITOR
021516 012746 021524                    CLR     -(SP)                          ;;INSURE THE 'T' BIT IS CLEAR
021522 000427                            MOV     #$CLR.T,-(SP)                 ;;SETUP FOR AN RTI OR RTT
BR      $RTRN                            ;;GO DO AN RTI OR RTT TO LOAD THE PSW
;;WITH A CLEARED 'T' BIT

```


021524
021524 013700 000042
021530 001405
021532 000005
021534 004710
021536 000240

\$CLR.T: MOV @#42,RO
BEQ \$DOAGN
RESET
\$ENDAD: JSR PC,(RO)
NOP

::INSURE RO CONTAINS THE MONITORS
::RETURN ADDRESS
::CLEAR THE WORLD
::GO TO MONITOR
::SAVE ROOM

```

021540 000240          NOP          ;;FCR
021542 000240          NOP          ;;ACT11
021544          SDOAGN:
021544 013746 177776    MOV @#PS,-(SP)  ;;PUT THE PS ON THE STACK AND
021550 042716 000020    BIC #20,(SP)  ;;CLEAR THE 'T' BIT
021554 032777 010000 157366 BIT #BIT12,@SWR  ;;&& RUN WITH TRACE TRAP?
021562 001005          BNE 1$      ;;BR IF NO
021564 005137 021610    COM $TBIT    ;;IS IT TIME FOR TRACE TRAP
021570 100402          BMI 1$      ;;BR IF NO
021572 052716 000020    BIS #20,(SP)  ;;SET TRACE TRAP
021576 012746 021604    1$: MOV #SLOOP,-(SP) ;;JUMP TO START OF TEST
021602 000002          SRTN: RTI     ;;RETURN--THIS IS CHANGED TO
                                           ;;AN 'RTT' IF 'RTT' IS A LEGAL
                                           ;;INSTRUCTION

021604          $LOOP:
021604 000137 010510    JMP @#LOOP    ;;RETURN
021610 000000          $TBIT: .WORD 0      ;;'T' BIT STATE INDICATOR
021612 377 377 000 $ENULL: .BYTE -1,-1,0 ;;NULL CHARACTER STRING
                                           .EVEN
    
```

2863
2864
2865
2866
2867
2868
2869
2870 021616 105737 020034
2871 021622 001411
2872 021624 105737 020035
2873 021630 100006
2874 021632 033715 006154
2875 021636 001403
2876 021640 011537 001174
2877 021644 000402
2878 021646 062716 000002
2879 021652 000207
2880
2881
2882
2883
2884
2885 021654 105737 020034
2886 021660 001411
2887 021662 105737 020035
2888 021666 100006
2889 021670 033715 006154
2890 021674 001003
2891 021676 011537 001174
2892 021702 000402
2893 021704 062716 000002
2894 021710 000207
2895
2896

```
.SBTTL SUBROUTINES UNIQUE TO TEST #7 ONLY
*****
*****
*
* THIS SUBROUTINE CHECKS THE STATUS OF THE BIT POINTED TO IN FLOATR
* IN THE LOCATION POINTED TO BY R5 (EITHER $DDW0 OR $DDW1), AND DETER-
* MINES IF THE LOCATION SHOULD BE DISABLED.
DSABLD: TSTB $ENV ;TEST APT STATUS
        BEQ 1$ ;BRANCH IF NOT APT
        TSTB $ENVM ;DOES APT SAY TO SIZE
        BPL 1$ ;BRANCH TO EXIT IF NOT
        BIT FLOATR,(R5) ;TEST DISABLE STATUS
        BEQ 1$ ;BRANCH IF IT SHOULD BE DISABLED
        MOV (R5),$TMP1 ;MOVE CONTENTS OF DEVICE DESCRIPTOR WORD TO $TMP1
        BR 2$ ;BRANCH OVER RETURN CORRECTION
1$: ADD #2,(SP) ;CHOCOLATE FUDGE RETURN OVER ERROR CALL
2$: RTS PC ;RETURN WITHOUT CORRECTING STACK SO ERROR WILL CALL
*****
*****
* THIS SUBROUTINE CHECKS THE STATUS OF THE BIT POINTED TO IN FLOATR
* IN THE LOCATION POINTED TO BY R5 (EITHER $DDW0 OR $DDW1), AND DETER-
* MINES IF THE LOCATION SHOULD BE ENABLED.
ENABLD: TSTB $ENV ;TEST APT STATUS
        BEQ 1$ ;BRANCH IF NOT APT
        TSTB $ENVM ;DOES APT SAY TO SIZE
        BPL 1$ ;BRANCH TO EXIT IF NOT
        BIT FLOATR,(R5) ;TEST DISABLE STATUS
        BNE 1$ ;BRANCH IF IT SHOULD BE ENABLED
        MOV (R5),$TMP1 ;MOVE CONTENTS OF DEVICE DESCRIPTOR WORD TO $TMP1
        BR 2$ ;BRANCH OVER RETURN CORRECTION
1$: ADD #2,(SP) ;VANILLA FUDGE RETURN OVER ERROR CALL
2$: RTS PC ;RETURN WITHOUT CORRECTING STACK SO ERROR WILL CALL
*****
*****
```

```

2898 .SBTTL ASCII AND ERROR MESSAGES
2899 .RADIX 2 ;THIS ENABLES YOU TO SEE THE BIT PATTERNS BELOW
2900 021712 177777 PATRNS: .WORD 1111111111111111 ;ALL BITS SET
2901 021714 000000 .WORD 0000000000000000 ;ALL BITS CLEAR
2902 021716 125252 .WORD 1010101010101010 ;ODD BITS SET, EVEN BITS CLEAR
2903 021720 052525 .WORD 0101010101010101 ;EVEN BITS SET, ODD BITS CLEAR
2904 021722 031463 .WORD 0011001100110011 ;ALTERNATING PAIRS OF BITS SET
2905 021724 007417 .WORD 0000111100001111 ;ALTERNATING GROUPS OF 4 BITS SET
2906 021726 000377 .WORD 0000000011111111 ;LOWER BYTE SET, UPPER BYTE CLEAR
2907 021728 000010 .RADIX 8 ;THIS RETURNS MODE BACK TO OCTAL
2908 021730 000 000 DFMSG: .BYTE 0,0,0,0,0 ;ALL NUMBERS ARE TO BE PRINTED IN OCTAL
2909 021735 200 125 UBMAVA: .ASCIZ <CRLF>?UNIBUS MEMORY AVAILABLE = ?
2910 021771 040 113 UBMEND: .ASCIZ ? K?<CRLF>

```

2911	021775	104	111	101	CPUMSG:	.ASCIZ	?DIAGNOSTIC HAS DETERMINED THAT CPU IS AN 11/?
2912	022052	062	064	200	ELEV24:	.ASCIZ	?24?<CRLF>
2913	022056	064	064	200	ELEV44:	.ASCIZ	?44?<CRLF>
2914	022062	116	117	124	EM1:	.ASCIZ	?NOT THE CORRECT TRAP CONDITION THROUGH ERRVEC (#004)?
2915	022147	125	116	105	EM2:	.ASCIZ	?UNEXPECTED CPU TRAP THROUGH ERRVEC (#004)?
2916	022221	125	116	105	EM3:	.ASCIZ	?UNEXPECTED MEMORY MANAGEMENT TRAP, MEMORY MANAGEMENT STATUS REGISTERS?
2917	022327	123	125	115	EM4:	.ASCIZ	?SUMMARY OF MAP REGISTERS THAT TIMED OUT ON READ?
2918	022407	123	125	115	EM5:	.ASCIZ	?SUMMARY OF DUAL ADDRESSING ERRORS ON LOADING MAP REGISTERS?
2919	022502	123	125	115	EM6:	.ASCIZ	?SUMMARY OF BIT PATTERN FAILURES IN LOWER 16 BITS OF MAP REGISTERS?
2920	022604	123	125	115	EM7:	.ASCIZ	?SUMMARY OF BIT PATTERN FAILURES IN UPPER 6 BITS OF MAP REGISTERS?
2921	022705	103	101	116	EM10:	.ASCII	?CAN'T GET TO MAIN MEMORY FROM UNIBUS WITH THE MAP OFF?<CRLF>
2922	022773	123	117	040		.ASCIZ	?SO I'LL JUMP TO THE SIZE JUMPER TEST FOR VERIFICATION?
2923	023061	123	125	115	EM11:	.ASCIZ	?SUMMARY OF COUNT PATTERN FAILURES ON THE UNIBUS DATA PATH?
2924	023153	125	116	111	EM12:	.ASCIZ	?UNIBUS MAP IS RELOCATING WHEN NOT ENABLED?
2925	023225	103	101	116	EM13:	.ASCII	?CANNOT USE ANY OF THE MAP REGISTERS OR PHYSICAL?<CRLF>
2926	023305	101	104	104		.ASCII	?ADDRESS BIT14 IS STUCK LOW, MUST RESTART PROGRAM?<CRLF>
2927	023366	111	106	040		.ASCIZ	?IF YOU DON'T LOOP ON THIS PROBLEM.?
2928	023431	123	125	115	EM14:	.ASCIZ	?SUMMARY OF UNIBUS ADDRESS ERRORS, WITH THE MAP RELOCATION DISABLED?
2929	023534	115	101	111	EM15:	.ASCIZ	?MAIN MEMORY TIMEOUT OVER THE UNIBUS DID NOT OCCUR PROPERLY?
2930	023627	122	105	114	EM16:	.ASCIZ	?RELOCATION THROUGH THE MAP WAS NOT CORRECT, CARRY PROPAGATION?
2931	023725	116	117	040	EM17:	.ASCIZ	?NO UNIBUS MEMORY EXISTS?
2932	023755	111	116	124	EM20:	.ASCIZ	?INTERRUPT/ABORT LOGIC TESTS TRAP TO LOCATION 114 DID NOT OCCUR?
2933	024054	111	116	124	EM21:	.ASCIZ	?INTERRUPT/ABORT TESTS R4 WAS OVERWRITTEN WITH?
2934	024132	111	116	124	EM22:	.ASCIZ	?INTERRUPT/ABORT TESTS TRAP DID NOT OCCUR DUE TO ABORT?
2935	024220	114	115	101	FM23:	.ASCIZ	?LMA NOT LOADED PROPERLY?
2936	024250	114	115	101	EM24:	.ASCIZ	?LMA FORCE JUMPER BIT NOT ZERO?
2937	024306	114	115	101	EM25:	.ASCIZ	?LMA FORCE JUMPER BIT NOT SET?
2938	024343	114	115	101	EM26:	.ASCIZ	?LMA CONTROL BITS INCORRECT?
2939	024376	106	117	122	EM27:	.ASCIZ	?FORCE JUMPER BIT FAILS TO REVERT MAP REGISTER STATUS TO DEFAULT?
2940	024476	113	111	120	EM30:	.ASCIZ	?KIPAR5 NOT LOADED PROPERLY - SKIPPING NEXT TEST?
2941	024556	124	110	105	EM201:	.ASCIZ	?THE FOLLOWING REGISTERS TIMED OUT WHEN READ?
2942	024632	124	110	105	EM202:	.ASCIZ	?THE FOLLOWING ARE DUAL ADDRESSING ERRORS IN THE UNIBUS MAP?
2943	024725	124	110	105	EM203:	.ASCIZ	?THE BIT PATTERN THROUGH THE MAP REGISTERS FAILED?
2944	025006	125	116	111	EM204:	.ASCIZ	?UNIBUS DATA PATH COUNT PATTERN FAILURE?
2945	025055	125	116	111	EM205:	.ASCIZ	?UNIBUS ADDRESSING ERRORS, MAP RELOCATION DISABLED?
2946	025137	104	101	124	EM206:	.ASCIZ	?DATA PATTERN NOT CORRECT?
2947	025170	122	105	106	EM207:	.ASCIZ	?REFERENCED MAP REGISTER 0 WITH ADDRESS ONE BIT DIFFERENT THAN 17770200?
2948	025277	115	101	120	EM210:	.ASCIZ	?MAP REGISTER(S) UNDER TEST DID NOT RESPOND IN DUAL MAPPING TEST?
2949	025377	122	105	114	EM211:	.ASCII	?RELOCATION THROUGH THE MAP WAS NOT CORRECT, CARRY PROPAGATION?<CRLF>
2950	025475	124	105	123		.ASCIZ	?TEST CODE BEING RUN OVER UNIBUS?
2951	025535	115	101	111	EM212:	.ASCII	?MAIN MEMORY TIMEOUT OVER THE UNIBUS DID NOT OCCUR PROPERLY?<CRLF>
2952	025630	124	105	123		.ASCIZ	?TEST CODE BEING RUN OVER UNIBUS?
2953	025670	115	101	120	EM213:	.ASCIZ	?MAP REGISTER ENABLED WHEN DDW SAYS IT SHOULD BE DISABLED?
2954	025761	115	101	120	EM214:	.ASCIZ	?MAP REGISTER DISABLED WHEN DDW SAYS IT SHOULD BE ENABLED?

Line	Address	Length	Offset	Label	Format	Fields
2955				.SBTTL	DATA HEADERS	
2956	026052	122	105	DH1:	.ASCIZ	?RECEIVD EXPECTD TESTNO ERR PC?
2957	026111	122	105	DH2:	.ASCIZ	?RECEIVD TESTNO ERR PC?
2958	026140	123	101	DH3:	.ASCII	?STATUS AUTOI/D VIRTUAL?<CRLF>
2959	026170	122	105		.ASCIZ	?REGISTR REGISTR ADDRESS TESTNO ERR PC?
2960	026237	122	105	DH4:	.ASCII	?REGADRS REGADRS?<CRLF>
2961	026261	040	042		.ASCIZ	? "OR" "AND" #ERRORS TESTNO ERR PC?
2962	026334	122	105	DH5:	.ASCII	?REGLOAD REGLOAD REGDUAL REGDUAL?<CRLF>
2963	026402	040	042		.ASCIZ	? "OR" "AND" "OR" "AND" #ERRORS TESTNO?
2964	026471	115	101	DH6:	.ASCII	?MAPREG MAPREG EXPECTD EXPECTD RECEIVD RECEIVD?<CRLF>
2965	026555	040	042		.ASCIZ	? "OR" "AND" "OR" "AND" #ERRORS TESTNO?
2966	026660	124	105	DH10:	.ASCIZ	?TESTNO ERR PC?
2967	026677	105	130	DH11:	.ASCII	?EXPECTD EXPECTD RECEIVD RECEIVD?<CRLF>
2968	026745	040	042		.ASCIZ	? "OR" "AND" "OR" "AND" #ERRORS TESTNO?
2969	027034	103	117	DH15:	.ASCII	?CONDITN CONDITN?<CRLF>
2970	027054	105	130		.ASCIZ	?EXPECTD RECEIVD TESTNO ERR PC?
2971	027113	124	105	DH23:	.ASCIZ	?TESTNO ERR PC LMAEXP LMARCV?
2972	027154	124	105	DH24:	.ASCIZ	?TESTNO ERR PC LMAEXP LMARCV?
2973	027213	124	105	DH27:	.ASCIZ	?TESTNO ERR PC LMARCV KIPAR4?
2974	027252	124	105	DH30:	.ASCIZ	?TESTNO ERR PC PR5EXP PR5RCV?
2975	027311	122	105	DH201:	.ASCIZ	?REGADRS TESTNO ERR PC?
2976	027340	115	101	DH202:	.ASCII	?MAPREG MAPREG NON-ZER?<CRLF>
2977	027374	124	105		.ASCIZ	?TESTING DUALED CONTNTS TESTNO ERR PC?
2978	027447	122	105	DH203:	.ASCIZ	?REGADRS PATTRN EXPCTD RECEVD TESTNO ERR PC?
2979	027530	105	130	DH204:	.ASCIZ	?EXPECTD RECEIVD ADDRSLD TESTNO ERR PC?
2980	027601	101	104	DH205:	.ASCII	?ADDRESS ADDRESS?<CRLF>
2981	027623	105	130		.ASCIZ	?EXPECTD RECEIVD TESTNO ERR PC?
2982	027666	101	104	DH206:	.ASCIZ	?ADDRESS EXPCTD RECVD TESTNO ERR PC?
2983	027736	101	104	DH207:	.ASCIZ	?ADDRUSED BITDIFF TESTNO ERR PC?
2984	027777	124	105	DH210:	.ASCIZ	?TESTNO ERR PC MAPREGADR?
2985	030031	103	117	DH211:	.ASCII	?CORRECT EXPECTD RECEIVD?<CRLF>
2986	030063	101	104		.ASCIZ	?ADDRESS DATA FROM UB TESTNO ERR PC?
2987	030134	103	117	DH212:	.ASCII	?CONDITN CONDITN?<CRLF>
2988	030154	105	130		.ASCIZ	?EXPECTD RECEIVD TESTNO ERR PC?
2989	030213	124	105	DH213:	.ASCIZ	?TESTNO ERR PC REG NO DDWDAT DDWADR?
2990					.EVEN	

Line	Address	Offset	Value	Label	Symbol
2991				.SBTTL	DATA TABLES
2992	030262	001314	001312	001310	DT1: .WORD PCPUER,CPUEXP,TESTNO,BADPC,0
2993	030274	001314	001310	001324	DT2: .WORD PCPUER,TESTNO,BADPC,0
2994	030304	001334	001336	001340	DT3: .WORD PMMR0,PMMR1,PMMR2,TESTNO,BADPC,0
2995	030320	001234	001230	001302	DT4: .WORD ADDROR,ADRAND,ERRCNT,TESTNO,\$ERRPC,0
2996	030334	001234	001230	001244	DT5: .WORD ADDROR,ADRAND,DATAOR,DATAND,ERRCNT,TESTNO,0
2997	030352	001234	001230	001252	DT6: .WORD ADDROR,ADRAND,PATTOR,PATAND,DATAOR,DATAND,ERRCNT,TESTNO,0
2998	030374	001310	001114	000000	DT10: .WORD TESTNO,\$ERRPC,0
2999	030402	001252	001250	001244	DT11: .WORD PATTOR,PATAND,DATAOR,DATAND,ERRCNT,TESTNO,0
3000	030420	001234	001230	001244	DT14: .WORD ADDROR,ADRAND,DATAOR,DATAND,ERRCNT,TESTNO,0
3001	030436	001312	001314	001310	DT15: .WORD CPUEXP,PCPUER,TESTNO,\$ERRPC,0
3002	030450	001310	001114	006144	DT23: .WORD TESTNO,\$ERRPC,EADRES,EADRS2,0
3003	030462	001310	001114	001160	DT24: .WORD TESTNO,\$ERRPC,\$REG1,LMAHI,0
3004	030474	001310	001114	001172	DT26: .WORD TESTNO,\$ERRPC,\$TMP0,\$REG2,0
3005	030506	001310	001114	001172	DT27: .WORD TESTNO,\$ERRPC,\$TMP0,KIPAR4,0
3006	030520	001310	001114	001204	DT30: .WORD TESTNO,\$ERRPC,\$TMP5,KIPAR5,0
3007	030532	006144	001310	001114	DT201: .WORD EADRES,TESTNO,\$ERRPC,0
3008	030542	006150	006144	001200	DT202: .WORD EADRS2,EADRES,\$TMP3,TESTNO,\$ERRPC,0
3009	030556	006150	001172	001166	DT203: .WORD EADRS2,\$TMP0,\$REG4,\$REG3,TESTNO,\$ERRPC,0
3010	030574	001172	001174	001162	DT204: .WORD \$TMP0,\$TMP1,\$REG2,TESTNO,\$ERRPC,0
3011	030610	006144	006150	001310	DT205: .WORD EADRES,EADRS2,TESTNO,\$ERRPC,0
3012	030622	006144	001202	001204	DT206: .WORD EADRES,\$TMP4,\$TMP5,TESTNO,\$ERRPC,0
3013	030636	006144	001156	001310	DT207: .WORD EADRES,\$REG0,TESTNO,\$ERRPC,0
3014	030650	001310	001114	006144	DT210: .WORD TESTNO,\$ERRPC,EADRES,0
3015	030660	006144	001164	001162	DT211: .WORD EADRES,\$REG3,\$REG2,TESTNO,\$ERRPC,0
3016	030674	001312	001314	001310	DT212: .WORD CPUEXP,PCPUER,TESTNO,\$ERRPC,0
3017	030706	001310	001114	001172	DT213: .WORD TESTNO,\$ERRPC,\$TMP0,\$TMP1,\$REG5,0

DATA FIELDS

3018				.SBTTL	DATA FIELDS	
3019	030722	000	000	000	DF1:	.BYTE 0,0,0,0,0
3020	030727	002	002	001	DF4:	.BYTE 2,2,1,0,0
3021	030734	002	002	002	DF5:	.BYTE 2,2,2,2,1,0
3022	030742	002	002	000	DF6:	.BYTE 2,2,0,0,0,0,1,0
3023	030752	000	000	000	DF11:	.BYTE 0,0,0,0,1,0
3024	030760	002	002	000	DF14:	.BYTE 2,2,0,0,1,0
3025	030766	000	000	002	DF23:	.BYTE 0,0,2,2
3026	030772	002	000	000	DF201:	.BYTE 2,0,0,0,0,0
3027	031000	000	000	003	DF204:	.BYTE 0,0,3,0,0
3028	031005	000	000	002	DF210:	.BYTE 0,0,2
3029	000001					.END

ABASE = 000000
ACDW1 = 000000
ACDW2 = 000000
ACPUOP = 000000
ADDROR = 001234
ADDW0 = 177777
ADDW1 = 177777
ADDW10 = 000000
ADDW11 = 000000
ADDW12 = 000000
ADDW13 = 000000
ADDW14 = 000000
ADDW15 = 000000
ADDW2 = 000000
ADDW3 = 000000
ADDW4 = 000000
ADDW5 = 000000
ADDW6 = 000000
ADDW7 = 000000
ADDW8 = 000000
ADDW9 = 000000
ADEVCT = 000000
ADEVM = 000000
ADRAND = 001230
ADREXT = 005250
AENV = 000000
AENVM = 000000
AFATAL = 000000
AMADR1 = 000000
AMADR2 = 000000
AMADR3 = 000000
AMADR4 = 000000
AMAMS1 = 000000
AMAMS2 = 000000
AMAMS3 = 000000
AMAMS4 = 000000
AMSGAD = 000000
AMSGLG = 000000
AMSGTY = 000000
AMTYP1 = 000000
AMTYP2 = 000000
AMTYP3 = 000000
AMTYP4 = 000000
APASS = 000000
APRIOR = 000000
APTCSU = 000040
APTENV = 000001
APTSIZ = 000200
APTSPO = 000100
ASWREG = 000000
ATESTN = 000000

BIT0 = 000001
BIT00 = 000001
BIT01 = 000002
BIT02 = 000004
BIT03 = 000010
BIT04 = 000020
BIT05 = 000040
BIT06 = 000100
BIT07 = 000200
BIT08 = 000400
BIT09 = 001000
BIT1 = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2 = 000004
BIT3 = 000010
BIT4 = 000020
BIT5 = 000040
BIT6 = 000100
BIT7 = 000200
BIT8 = 000400
BIT9 = 001000
BPTVEC = 000014
BUPWIN = 001274
CACHE = 177746
CACHEV = 000114
CHARCT = 006140
CLRMAP = 004352
CMPE = 177744
CNTR = 001304
CONTRL = 177746
CPFLAG = 004604
CPUER = 004602
CPUERR = 177766
CPUEXP = 001312
CPUMSG = 021775
CPUTYP = 007774
CR = 000015
CRLF = 000200
CTRAPS = 000116
CTRAPV = 000114
DATA = 001350
DATAND = 001240
DATAOR = 001244
DATEXT = 005200
DATO = 100000
DATOB = 140000

DF210 = 031005
DF23 = 030766
DF4 = 030727
DF5 = 030734
DF6 = 030742
DH1 = 026052
DH10 = 026660
DH11 = 026677
DH15 = 027034
DH2 = 026111
DH201 = 027311
DH202 = 027340
DH203 = 027447
DH204 = 027530
DH205 = 027601
DH206 = 027666
DH207 = 027736
DH210 = 027777
DH211 = 030031
DH212 = 030134
DH213 = 030213
DH23 = 027113
DH24 = 027154
DH27 = 027213
DH3 = 026140
DH30 = 027252
DH4 = 026237
DH5 = 026334
DH6 = 026471
DISPLA = 001152
DSABLD = 021616
DTMS = 006120
DTMSG = 006124
DT1 = 030262
DT10 = 030374
DT11 = 030402
DT14 = 030420
DT15 = 030436
DT2 = 030274
DT201 = 030532
DT202 = 030542
DT203 = 030556
DT204 = 030574
DT205 = 030610
DT206 = 030622
DT207 = 030636
DT210 = 030650
DT211 = 030660
DT212 = 030674
DT213 = 030706
DT23 = 030450

DT5 = 030334
DT6 = 030352
EADRES = 006144
EADRS2 = 006150
ELEV24 = 022052
ELEV44 = 022056
EMTVEC = 000030
EM1 = 022062
EM10 = 022705
EM11 = 023061
EM12 = 023153
EM13 = 023225
EM14 = 023431
EM15 = 023534
EM16 = 023627
EM17 = 023725
EM2 = 022147
EM20 = 023755
EM201 = 024556
EM202 = 024632
EM203 = 024725
EM204 = 025006
EM205 = 025055
EM206 = 025137
EM207 = 025170
EM21 = 024054
EM210 = 025277
EM211 = 025377
EM212 = 025535
EM213 = 025670
EM214 = 025761
EM22 = 024132
EM23 = 024220
EM24 = 024250
EM25 = 024306
EM26 = 024343
EM27 = 024376
EM3 = 022221
EM30 = 024476
EM4 = 022327
EM5 = 022407
EM6 = 022502
EM7 = 022604
ENABLD = 021654
ERRCNT = 001302
ERROR = 104000
ERRVEC = 000004
ERTYPE = 002012
ER200 = 001652
EXTOUT = 006142
FJBIT = 000100

HITMIS = 177752
HT = 000011
IOTVEC = 000020
JMPMSG = 006422
KDPAR0 = 172360
KDPAR1 = 172362
KDPAR2 = 172364
KDPAR3 = 172366
KDPAR4 = 172370
KDPAR5 = 172372
KDPAR6 = 172374
KDPAR7 = 172376
KDPDR0 = 172320
KDPDR1 = 172322
KDPDR2 = 172324
KDPDR3 = 172326
KDPDR4 = 172330
KDPDR5 = 172332
KDPDR6 = 172334
KDPDR7 = 172336
KERSTK = 001100
KIPAR0 = 172340
KIPAR1 = 172342
KIPAR2 = 172344
KIPAR3 = 172346
KIPAR4 = 172350
KIPAR5 = 172352
KIPAR6 = 172354
KIPAR7 = 172356
KIPDR0 = 172300
KIPDR1 = 172302
KIPDR2 = 172304
KIPDR3 = 172306
KIPDR4 = 172310
KIPDR5 = 172312
KIPDR6 = 172314
KIPDR7 = 172316
KSP = 2000006
LF = 000012
LMAHI = 177736
LMALOW = 177734
LOADRS = 177740
LOOP = 010510
LOWEST = 001254
LREGL = 001276
LREGU = 001300
MAINT = 177750
MAPADD = 005404
MAPH0 = 170202
MAPH00 = 170202
MAPH01 = 170206

AJNIT = 000000
AUSWR = 000000
AVECT1 = 000000
AVECT2 = 000000
BADCPU 007446
BADPC 001324

DFMSG 021730
DF1 030722
DF11 030752
DF14 030760
DF201 030772
DF204 031000

DT24 030462
DT26 030474
DT27 030506
DT3 030304
DT30 030520
DT4 030320

K 12

FLAG 001306
FLOATR 006154
FTTHRU 005120
GMRMOD 007702
HIADRS = 177742
HIGEST 001256

MAPH02 = 170212
MAPH03 = 170216
MAPH04 = 170222
MAPH05 = 170226
MAPH06 = 170232
MAPH07 = 170236

SEQ 0153

MAPH1 = 170206
MAPH10= 170242
MAPH11= 170246
MAPH12= 170252
MAPH13= 170256
MAPH14= 170262
MAPH15= 170266
MAPH16= 170272
MAPH17= 170276
MAPH2 = 170212
MAPH20= 170302
MAPH21= 170306
MAPH22= 170312
MAPH23= 170316
MAPH24= 170320
MAPH25= 170326
MAPH26= 170332
MAPH27= 170336
MAPH3 = 170216
MAPH30= 170342
MAPH31= 170346
MAPH32= 170352
MAPH33= 170356
MAPH34= 170362
MAPH35= 170366
MAPH36= 170372
MAPH37= 170376
MAPH4 = 170222
MAPH5 = 170226
MAPH6 = 170232
MAPH7 = 170236
MAPL0 = 170200
MAPLU0= 170200
MAPL01= 170204
MAPL02= 170210
MAPL03= 170214
MAPL04= 170220
MAPL05= 170224
MAPL06= 170230
MAPL07= 170234
MAPL1 = 170204
MAPL10= 170240
MAPL11= 170244
MAPL12= 170250
MAPL13= 170254
MAPL14= 170260
MAPL15= 170264
MAPL16= 170270
MAPL17= 170274
MAPL2 = 170210
MAPL20= 170300

MAPL27= 170334
MAPL3 = 170214
MAPL30= 170340
MAPL31= 170344
MAPL32= 170350
MAPL33= 170354
MAPL34= 170360
MAPL35= 170364
MAPL36= 170370
MAPL37= 170374
MAPL4 = 170220
MAPL5 = 170224
MAPL6 = 170230
MAPL7 = 170234
MASK1 006156
MASK2 006160
MEMERR= 177744
MFPT = 000007
MMFLAG 004746
MMRHI 001266
MMRLOW 001264
MMRO = 177572
MMR1 = 177574
MMR2 = 177576
MMR3 = 172516
MMTOTM 014420
MMTRAP 004744
MMVEC = 000250
MRQUES 007613
NEXMEM= 000040
NEXT 005140
NOMORE 007260
NXTTST 001346
OLDPC 001326
OLDPS 001330
OLDPSW 001332
PADRSH 001226
PADRSL 001224
PATAND 001250
PATEXT 005224
PATRNS 021712
PATTOR 001252
PCONTR 001320
PCPUER 001314
PIRQ = 177772
PIRQVE= 000240
PMAINT 001322
PMMR0 001334
PMMR1 001336
PMMR2 001340
PPARER 001316

PR5 = 000240
PR6 = 000300
PR7 = 000340
PS = 177776
PSW = 177776
PWRMSG 004122
PWRVEC= 000024
RELC22 011600
RESREG= 104414
RESVEC= 000010
RETRY 001344
RSIZE 001342
R10 =%000000
R11 =%000001
R12 =%000002
R13 =%000003
R14 =%000004
R15 =%000005
R3STAK 005530
R6 =%000006
R7 =%000007
SAVREG= 104412
SCOPE = 000004
SDPAR0= 172260
SDPAR1= 172262
SDPAR2= 172264
SDPAR3= 172266
SDPAR4= 172270
SDPAR5= 172272
SDPAR6= 172274
SDPAR7= 172276
SDPDR0= 172220
SDPDR1= 172222
SDPDR2= 172224
SDPDR3= 172226
SDPDR4= 172230
SDPDR5= 172232
SDPDR6= 172234
SDPDR7= 172236
SIPAR0= 172240
SIPAR1= 172242
SIPAR2= 172244
SIPAR3= 172246
SIPAR4= 172250
SIPAR5= 172252
SIPAR6= 172254
SIPAR7= 172256
SIPDR0= 172200
SIPDR1= 172202
SIPDR2= 172204
SIPDR3= 172206

SIZELO= 177760
SPECST 006162
SPSUPP 002413
SR0 = 177572
SR1 = 177574
SR2 = 177576
SR3 = 172516
SSP =%000006
STACK = 001100
START 010000
STKLMT= 177774
SUPSTK= 000700
SWR 001150
SW0 = 000001
SW00 = 000001
SW01 = 000002
SW02 = 000004
SW03 = 000010
SW04 = 000020
SW05 = 000040
SW06 = 000100
SW07 = 000200
SW08 = 000400
SW09 = 001000
SW1 = 000002
SW10 = 002000
SW11 = 004000
SW12 = 010000
SW13 = 020000
SW14 = 040000
SW15 = 100000
SW2 = 000004
SW3 = 000010
SW4 = 000020
SW5 = 000040
SW6 = 000100
SW7 = 000200
SW8 = 000400
SW9 = 001000
SYSTID= 177764
TBIT = 000020
TBITO = 104416
TBITOF 004306
TBITR = 104420
TBITRE 004334
TBITVE= 000014
TCPMRA 005632
TESTNO 001310
TIMEOU 004410
TIMOUT= 000020
TKVEC = 000060

TST1 010600
TST10 013440
TST11 014130
TST12 014362
TST13 014544
TST14 014574
TST15 014674
TST16 014722
TST17 014760
TST2 011022
TST20 015010
TST21 015352
TST22 015706
TST23 016112
TST24 016152
TST25 016352
TST26 016416
TST27 016616
TST3 011400
TST30 017010
TST31 017202
TST4 011710
TST5 012130
TST6 012470
TST7 012604
TYPDAT= 002214
TYPDS = 104410
TYPE = 104400
TYPOC = 104402
TYPON = 104406
TYPOS = 104404
TYPVAD 002414
UBADDR 002522
UBMAVA 021735
UBMEND 021771
UBMHI 001262
UBMLOW 001260
UBRHI 001272
UBRLOW 001270
UDPAR0= 177660
UDPAR1= 177662
UDPAR2= 177664
UDPAR3= 177666
UDPAR4= 177670
UDPAR5= 177672
UDPAR6= 177674
UDPAR7= 177676
UDPDR0= 177620
UDPDR1= 177622
UDPDR2= 177624
UDPDR3= 177626

MAPL21= 170304
MAPL22= 170310
MAPL23= 170314
MAPL24= 170320
MAPL25= 170324
MAPL26= 170330

PRETST 017662
PR0 = 000000
PR1 = 000040
PR2 = 000100
PR3 = 000140
PR4 = 000200

SIPDR4= 172210
SIPDR5= 172212
SIPDR6= 172214
SIPDR7= 172216
SIZEHI= 177762
SIZEJ 012620

M 12

TOFLAG 004412
TOMSG 006262
TPVEC = 000064
TRAPVE= 000034
TRIVEC= 000014
TSTLOC 005046

UDPDR4= 177630
UDPDR5= 177632
UDPDR6= 177634
UDPDR7= 177636
UIPAR0= 177640
UIPAR1= 177642

SEQ 0155

UIPAR2= 177644	\$DB20 004166	\$FATAL 020016	\$NULL 001144	\$TKB 001136
UIPAR3= 177646	\$DDW0 020100	\$FFLG 021304	\$NWTST= 000001	\$TKS 001134
UIPAR4= 177650	\$DDW1 020102	\$FILLC 001146	\$OCNT 003426	\$TMP0 001172
UIPAR5= 177652	\$DDW10 020124	\$FILLS 001145	\$OCTVL 004270	\$TMP1 001174
UIPAR6= 177654	\$DDW11 020126	\$GDADR 001116	\$OMODE 003430	\$TMP2 001176
UIPAR7= 177656	\$DDW12 020130	\$GDDAT 001122	\$OVER 020502	\$TMP3 001200
UIPDR0= 177600	\$DDW13 020132	\$GET42 021506	\$PASS 020022	\$TMP4 001202
UIPDR1= 177602	\$DDW14 020134	\$HD = 000000	\$PASTM 020006	\$TMP5 001204
UIPDR2= 177604	\$DDW15 020136	\$HIBTS 020000	\$PWRAD 004110	\$TMP6 001206
UIPDR3= 177606	\$DDW2 020104	\$ICNT 001102	\$PWRDN 003766	\$TN = 000032
UIPDR4= 177610	\$DDW3 020106	\$ILLUP 004114	\$PWRMG 004104	\$TPB 001142
UIPDR5= 177612	\$DDW4 020110	\$ITEMB 001112	\$PWRUP 004034	\$TPFLG 001147
UIPDR6= 177614	\$DDW5 020112	\$LF 001222	\$QUES 001220	\$TPS 001140
UIPDR7= 177616	\$DDW6 020114	\$LFLG 021303	\$REGAD 001154	\$TRAP 003724
USESTK= 000600	\$DDW7 020116	\$LOOP 021604	\$REG0 001156	\$TRP = 000022
USP =%000006	\$DDW8 020120	\$LPADR 001104	\$REG1 001160	\$TRPAD 003744
\$APTHD 020000	\$DDW9 020122	\$LPERR 001106	\$REG2 001162	\$STM 020004
\$ATYC 021064	\$DEVCT 020024	\$MADR1 020046	\$REG3 001164	\$STNM 001100
\$ATY1 021040	\$DEVM 020072	\$MADR2 020052	\$REG4 001166	\$STPDS 003432
\$ATY3 021046	\$DOAGN 021544	\$MADR3 020056	\$REG5 001170	\$TYPE 002674
\$ATY4 021056	\$DTBL 003704	\$MADR4 020062	\$RESRE 002636	\$TYPEC 003050
\$BASE 020070	\$ENDAD 021534	\$MAIL 020014	\$RTRN 021602	\$TYPEX 003202
\$BDADR 001120	\$ENDCT 021354	\$MAMS1 020044	\$SAVRE 002600	\$TYPOC 003230
\$BDDAT 001124	\$ENULL 021612	\$MAMS2 020050	\$SAVR6 004120	\$TYPON 003244
\$BELL 001214	\$ENV 020034	\$MAMS3 020054	\$SCOPE 020140	\$TYPOS 003204
\$CDW1 020074	\$ENVM 020035	\$MAMS4 020060	\$SETUP= 000037	\$UNIT 020026
\$CDW2 020076	\$EOP 021306	\$MBADR 020002	\$SSWR 000176	\$UNITM 020010
\$CHARC 003200	\$EOPCT 021346	\$MFLG 021302	\$STUP = 177777	\$USWR 020040
\$CLR.T 021524	\$ERFLG 001101	\$MSGAD 020030	\$SVLAD 020446	\$VECT1 020064
\$CMTAG 001100	\$ERMAX 001113	\$MSGLG 020032	\$SVPC = 000204	\$VECT2 020066
\$CM1 = 000006	\$ERROR 020520	\$MSGTY 020014	\$SWR = 177400	\$XTSTR 020244
\$CM2 = 000014	\$ERRPC 001114	\$MTYP1 020045	\$SWREG 020036	\$GET4= 000001
\$CM3 = 000006	\$ERRTB 001352	\$MTYP2 020051	\$SWRMK= 000340	\$STRP = 000002
\$CM4 = 000007	\$ERTTL 001110	\$MTYP3 020055	\$TBIT 021610	\$OFILL 003427
\$CPUOP 020042	\$ESCAP 001212	\$MTYP4 020061	\$TESTN 020020	\$.SX = 020000
\$CRLF 001221	\$ETABL 020034	\$MXCNT 020516	\$TIMES 001210	\$.SY = 002012
\$DBLK 003714	\$ETEND 020140	\$NEXT 021360		

. ABS. 031010 000
 000000 001
 ERRORS DETECTED: 0

VIRTUAL MEMORY USED: 46872 WORDS (184 PAGES)
 DYNAMIC MEMORY: 20346 WORDS (78 PAGES)
 ELAPSED TIME: 00:08:38
 CKKUAB.BIN,CKKUAB.SEQ/-SP/NL:TOC=CKKUAB.MLB/ML,CKKUAB.P11